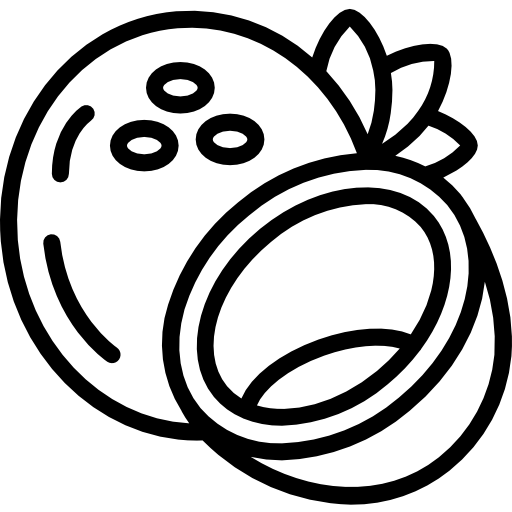
Министерство науки и высшего образования РФ

федеральное государственное автономное образовательное учреждение высшего образования

«Омский государственный технический университет»

Лицей Академии Яндекса

«Туристическое агентство Kokos»



Выполнили:

**Гончаров Пётр**

**Силантьев Алексей**

Омск, 2023

РЕФЕРАТ

Отчет 11 с., 2 рис., 1 табл., 3 источн., 1 прил.

ПРОЕКТ, GUI ИНТЕРФЕЙС, ДАННЫЕ, PYTHON, FLASK

Предмет исследования – WEB-приложение.

Цель исследования – разработка WEB-приложения для туристической

В процессе выполнения проекта мы получили множество новых навыков, которые пригодятся нам в будущем.

В результате выполнения проекта мы научились создавать WEB-приложения для организаций.

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ](#_2r9v0h46yy99) 3

[1 Описание проблемы и формулирование гипотезы для ее решения](#_ifzanbkzs7df) 5

[2 Проектирование приложения](#_pd7u93hyun9o) 5

[3 Реализация приложения](#_67z7petns5sz) 5

[ЗАКЛЮЧЕНИЕ](#_2et92p0) 6

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ](#_tyjcwt) 7

[ПРИЛОЖЕНИЕ А](#_1t3h5sf) Листинг 8

# ВВЕДЕНИЕ

Цель исследования – разработка приложения для реализации туров и привлечения клиентов приятным дизайном.

Для достижения поставленной цели требуется решить следующие задачи:

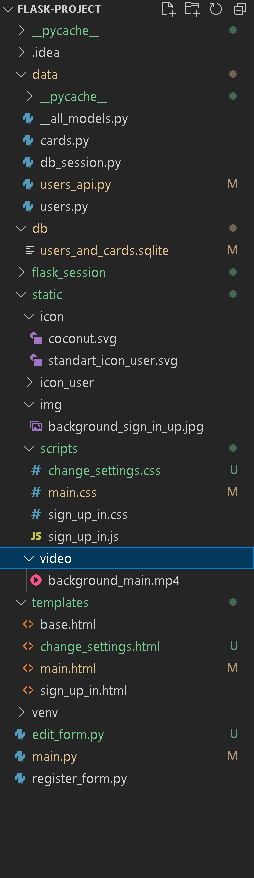
1. Сформулировать функции web-сайта
2. Определить типы страниц и способы их наполнения (Статически или динамически)
3. Создать само приложение

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Технологии сайта** | | | | | | | | |
| **requirements.txt** | **bootstrap** | **шаблоны** | **регистрация и авторизация** | **загрузка и использование файлов** | **API: REST или использование стороннего** | **Хранение данных (txt, csv или БД)** | **Хостинг** |
| **(railway)** |
| **+** | **+** | **+** | **Один тип пользователей:** | **Загрузка изображений для аватаров пользователей** | **Работа через http get запрос** | **БД SQLalchemy** | **+** |
| **Клиент** |
|  |
|  |  |
|  |  |

## 1 Описание проблемы и формулирование гипотезы для ее решения

*Реализация туров сейчас представляет из себя оффлайн обсуждение лично с клиентом, но такой метод не всегда удобен, поскольку есть множество людей с очень плотным графиком, которые хотят использовать отпуск по максимуму, и поэтому бронируют туры заранее. Именно на таких клиентов ориентировано наше WEB-приложение, которое будет предоставлять возможность бронировать путёвку и выбирать место, также прочитав отзывы, оставленные другими клиентами.*

## 2 Проектирование приложения

*Мы поняли, что финальная структура файлов будет выглядеть примерно так: *

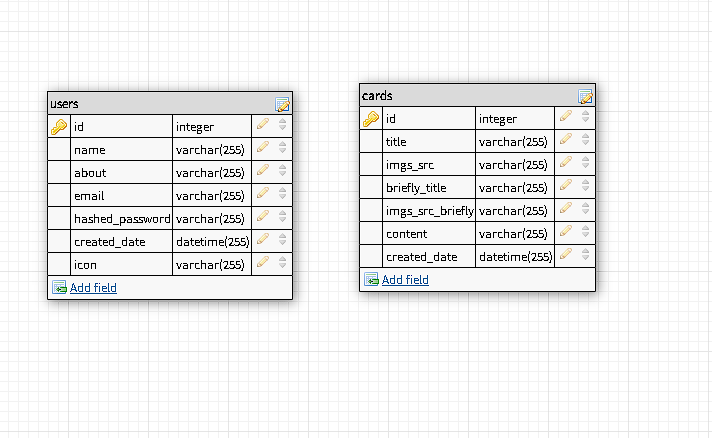
*Также мы поняли, что легче подгружать странички туров из БД нежели каждый раз отдельно добавлять туры.*

## 

## 3 Реализация приложения

*Сначала мы создали Flask-приложение, потом создали БД через Sqalchemy и в ней удобную структуру для добавления новых таблиц, потом создали подобие Rest API и отдельные blueprints для работы по http и json с основным flask-приложением, потом начали создавать странички и отдельные формы для регистраций, входа и т.д., затем мы начали оптимизировать странички под разные разрешения и внедрять WGSI сервер для адекватной работы с несколькими пользователями используя библиотеку waitress, затем мы задеплоили свой сервер на render.*

*Вот структура БД:*



# ЗАКЛЮЧЕНИЕ

*Текст заключения: на каждую задачу, прописанную во введении оформляете один абзац текста с подведением итогов по решению этой задачи. После абзацев по задачам идет абзац текста о дальнейших перспективах реализованного продукта.*

*Наше WEB-приложение станет посредником между клиентом и туристическим агентством.*

*У нас будет несколько типов страниц:*

* *Главная страница, на которой будет представлена основная информация.*
* *Страница регистрации пользователя.*
* *Страницы туров, которые будут обновляться динамически из баз данных.*

*Отрывок кода будет представлен в приложении.*

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. <https://www.youtube.com/>
2. <https://lyceum.yandex.ru/courses/766/groups/6082>
3. <https://chat.openai.com/chat>

# ПРИЛОЖЕНИЕ А

(обязательное)

**Листинг**

import flask

from flask\_login import LoginManager, login\_user, login\_required, logout\_user, current\_user

from flask import Flask, render\_template, url\_for, redirect, request, flash, make\_response, jsonify

import sqlalchemy

from register\_form import RegisterForm, LoginForm

from edit\_form import EditForm

from data import db\_session, users\_api, travels\_api

from data.users import User

from data.cards import Card

import requests

import os

from waitress import serve

app = Flask(\_\_name\_\_)

login\_manager = LoginManager()

login\_manager.init\_app(app)

app.config['SECRET\_KEY'] = 'yandexlyceum\_secret\_key'

@app.errorhandler(404)

def not\_found(error):

return make\_response(jsonify({'error': 'Not found'}), 404)

@app.errorhandler(400)

def bad\_request(\_):

return make\_response(jsonify({'error': 'Bad Request'}), 400)

@login\_manager.user\_loader

def load\_user(user\_id):

db\_sess = db\_session.create\_session()

return db\_sess.query(User).get(user\_id)

@app.route('/logout')

@login\_required

def logout():

logout\_user()

return redirect("/sign\_up\_in")

@app.route('/sign\_up\_in', methods=['GET', 'POST'])

def sign\_up\_in():

form\_login = LoginForm()

form\_registr = RegisterForm()

if form\_login.validate\_on\_submit() and request.form['submit'] == "Вход":

db\_sess = db\_session.create\_session()

user = db\_sess.query(User).filter(User.name == form\_login.username.data).first()

print(user)

if not user:

flash("Пользователь не найден", "error")

return render\_template('sign\_up\_in.html', title='Аутификация', form\_reg=form\_registr, form\_log=form\_login)

elif not user.check\_password(form\_login.password.data):

flash("Неправильный пароль", "error")

return render\_template('sign\_up\_in.html', title='Аутификация', form\_reg=form\_registr, form\_log=form\_login)

print(f"login: {form\_login.username.data, form\_login.password.data}")

login\_user(user, remember=True)

db\_sess.close()

return redirect("/main")

if (form\_registr.validate\_on\_submit() and request.form['submit'] == "Регестрация"):

print(f"registr: {form\_registr.email.data, form\_registr.username.data, form\_registr.password.data}")

try:

db\_sess = db\_session.create\_session()

if db\_sess.query(User).filter(User.name == form\_registr.username.data).first():

flash("Имя пользователя занято", "error")

return render\_template('sign\_up\_in.html', title='Аутификация', form\_reg=form\_registr, form\_log=form\_login)

requests.post(url="http://localhost:5000/api/users", json={"name": form\_registr.username.data, "password": form\_registr.password.data, "email": form\_registr.email.data, "about": ""}).json()

login\_user(db\_sess.query(User).filter(User.name == form\_login.username.data).first(), remember=True)

db\_sess.close()

return redirect("/main")

except requests.exceptions.JSONDecodeError:

flash("Такой email уже зарегистрирован", "error")

return render\_template('sign\_up\_in.html', title='Аутификация', form\_reg=form\_registr, form\_log=form\_login)

return redirect('/main')

if form\_registr.errors.get("email"):

if form\_registr.errors.get("email")[0] == "Неправильный email":

flash("Неправильный email", "error")

return render\_template('sign\_up\_in.html', title='Аутификация', form\_reg=form\_registr, form\_log=form\_login)

return render\_template('sign\_up\_in.html', title='Аутификация', form\_reg=form\_registr, form\_log=form\_login)

@app.route('/change\_settings', methods=['GET', 'POST'])

@login\_required

def change\_settings():

edit\_form = EditForm()

if edit\_form.validate\_on\_submit():

print(edit\_form.email.data, edit\_form.password.data, edit\_form.new\_password.data, edit\_form.about.data, edit\_form.photo.data)

respocde = requests.put(url="http://localhost:5000/api/users", json={"name": current\_user.name, "email": edit\_form.email.data, "password": edit\_form.password.data, "new\_password": edit\_form.new\_password.data, "about": edit\_form.about.data})

print(respocde.json())

if respocde.json().get("success") == "OK":

current\_user.email = edit\_form.email.data

current\_user.about = edit\_form.about.data

if edit\_form.photo.data.filename:

filename = edit\_form.photo.data.filename

print(filename)

edit\_form.photo.data.save(f"./static/icon/{filename}")

new\_filename = f"{current\_user.name}.jpg"

if os.path.exists(f"./static/icon\_user/{new\_filename}"):

os.remove(f'./static/icon\_user/{new\_filename}')

os.rename(f"./static/icon/{filename}", f"./static/icon\_user/{new\_filename}")

db\_sess = db\_session.create\_session()

user = db\_sess.query(User).filter(User.name == current\_user.name).one\_or\_none()

user.icon = url\_for("static", filename=f"icon\_user/{new\_filename}")

current\_user.icon = url\_for("static", filename=f"icon\_user/{new\_filename}")

db\_sess.commit()

flash("Изменения внесены", "success")

elif respocde.json().get("error") == "Invalid password":

flash("Неправильный пароль", "error")

elif respocde.json().get("error") == "User not found":

flash("Пользователь не найден", "error")

return render\_template("change\_settings.html", title="Настройки", form=edit\_form)

return render\_template("change\_settings.html", title="Настройки", form=edit\_form)

@app.route('/catalog/<int:travel\_id>')

@login\_required

def catalog\_page(travel\_id):

content = requests.get(url=f"http://localhost:5000/api/travels/{travel\_id}")

if content.json().get("error"):

return redirect("/main")

print(content.json())

return render\_template("travel\_card.html", title="Путешествие", travel\_info=content.json())

@app.route('/')

@app.route('/main')

def main\_page():

if current\_user.is\_authenticated:

return render\_template("main.html", title="Основная страница")

else:

return redirect("/sign\_up\_in")

def main():

db\_session.global\_init("db/users\_and\_cards.sqlite")

app.register\_blueprint(users\_api.blueprint)

app.register\_blueprint(travels\_api.blueprint)

app.run(port=5000, host='127.0.0.1')

#serve(app, host='127.0.0.1', port=5000)

if \_\_name\_\_ == '\_\_main\_\_':

main()