|  |  |
| --- | --- |
| Министерство образования Республики Беларусь  Учреждение образования «Полоцкий государственный университет» | |
|  | Факультет информационных технологий  Кафедра технологий программирования |
| Лабораторная работа №3 по курсу «Разработка и анализ требований»  «Учебное руководство по работе с API: Основы, интеграция и практика. Реализация API»  Вариант 5 | |
| Выполнил | Студент гр. 21-ИТ-1  Катушёнок И.В. |
| Проверил | Преподаватель  Васильева Д.М. |
| Полоцк, 2023г. | |

**Цель работы:** ознакомиться с понятием API (Application Programming Interface), понять, как оно используется, и предоставить примеры реализации API. Также, предоставить задание и контрольные вопросы для проверки понимания темы.

**Ход работы**

**Задание:**

1. Проектирование API

Напишите спецификацию RESTful API для простого блокнот-приложения, включая эндпоинты для создания, чтения, обновления и удаления заметок.

1. Использование API

Используя публичное API (например, JSONPlaceholder), создайте клиентское приложение, которое получает и отображает список пользователей. Bспользуйте библиотеку для HTTP-запросов (например, requests для Python).

1. Разработка RESTful API

Создать RESTful API, которое позволит управлять товарами в интернет-магазине. API должно предоставлять функциональность для создания, чтения, обновления и удаления товаров. Каждый товар должен иметь следующие характеристики: название, цена, описание и уникальный идентификатор(ID).

1. Интеграция с внешним API (ПО ВАРИАНТУ)

**Вариант 5**

Используйте CoinGecko API для создания приложения для отслеживания курсов криптовалют.

1. Аутентификация и авторизация

Расширьте функциональность вашего API(Задание 4), добавив аутентификацию и авторизацию пользователей. Реализуйте систему пользовательских аккаунтов и токенов доступа.

1. Реализация спецификации RESTful API для простого блокнот-приложения представлена в листинге 1.

Листинг 1 – Спецификации RESTful API для простого блокнот-приложения.

openapi: 3.0.0

info:

title: Блокнот-приложение

version: 1.0.0

servers:

- url: localhost:5000/notepad

paths:

/notes:

get:

summary: Получить список всех заметок

responses:

'200':

description: Успешный запрос

content:

application/json:

schema:

type: array

items:

$ref: '#/components/schemas/Note'

post:

summary: Создать новую заметку

requestBody:

required: true

content:

application/json:

schema:

$ref: '#/components/schemas/Note'

responses:

'201':

description: Заметка успешно создана

content:

application/json:

schema:

$ref: '#/components/schemas/Note'

/notes/{id}:

get:

summary: Получить информацию о заметке по идентификатору

parameters:

- name: id

in: path

required: true

schema:

type: string

responses:

'200':

description: Успешный запрос

content:

application/json:

schema:

$ref: '#/components/schemas/Note'

put:

summary: Обновить информацию о заметке

parameters:

- name: id

in: path

required: true

schema:

type: string

requestBody:

required: true

content:

application/json:

schema:

$ref: '#/components/schemas/Note'

responses:

'200':

description: Заметка успешно обновлена

content:

application/json:

schema:

$ref: '#/components/schemas/Note'

delete:

summary: Удалить заметку

parameters:

- name: id

in: path

required: true

schema:

type: string

responses:

'204':

description: Заметка успешно удалена

components:

schemas:

Note:

type: object

properties:

id:

type: string

title:

type: string

content:

type: string

required:

- title

- content

1. При выполнении данного задания было создано клиентское приложение, используя язык программирования Python, публичное API JSONPlaceholder и библиотеку для HTTP-запросов request, которое получает и отображает список пользователей. Реализация данного приложения представлена в листинге 2.

Листинг 2 – Реализация клиентского приложения, которое получает и отображает список пользователей.

import requests  
  
response = requests.get('https://jsonplaceholder.typicode.com/users')  
users = response.json()  
for user in users:  
 print(f"Имя: {user['name']}\t Email: {user['email']}")

Работа программы получения и отображения списка пользователей представлена рисунке 1.

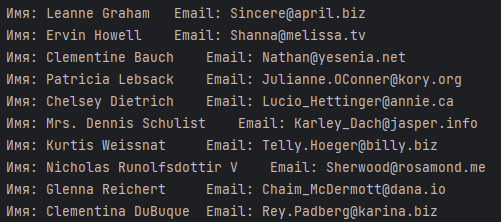


Рисунок 1 – Работа программы получения и отображения списка пользователей

1. При выполнении данного задания было реализовано RESTful API, которое позволит управлять товарами в интернет-магазине. Данное API было создано на языке программирования Python при помощи библиотеки Flask. Реализация данного приложения представлена в листинге 3.

Листинг 3 – Реализация клиентского приложения, которое позволит управлять товарами в интернет-магазине.

from flask import Flask, request, jsonify  
  
app = Flask(\_\_name\_\_)  
products = [  
 {'name':"батоны", 'price':'10000$', 'description':"свежие"}  
]  
  
@app.route('/products', methods=['POST'])  
def create\_product():  
 data = request.get\_json()  
 name = data.get('name')  
 price = data.get('price')  
 description = data.get('description')  
 product = {'name': name, 'price': price, 'description': description}  
 products.append(product)  
 return jsonify({'message': 'Product created successfully'}), 201  
  
@app.route('/products/<int:product\_id>', methods=['GET'])  
def get\_product(product\_id):  
 if product\_id >= len(products):  
 return jsonify({'error': 'Product not found'}), 404  
 return jsonify(products[product\_id])  
  
@app.route('/products/<int:product\_id>', methods=['PUT'])  
def update\_product(product\_id):  
 if product\_id >= len(products):  
 return jsonify({'error': 'Product not found'}), 404  
 data = request.get\_json()  
 products[product\_id]['name'] = data.get('name')  
 products[product\_id]['price'] = data.get('price')  
 products[product\_id]['description'] = data.get('description')  
 return jsonify({'message': 'Product updated successfully'})  
  
@app.route('/products/<int:product\_id>', methods=['DELETE'])  
def delete\_product(product\_id):  
 if product\_id >= len(products):  
 return jsonify({'error': 'Product not found'}), 404  
 del products[product\_id]  
 return jsonify({'message': 'Product deleted successfully'})  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 app.run(debug=True)

Работа программы, а именно запросы создания, чтения, обновления и удаления товаров были проверены при помощи HTTP-клиента для тестирования API Postman.

1. При выполнении данного задания была произведена интеграция с Coingecko API для создания приложения для получения и отслеживания курсов криптовалют. Реализация данного приложения представлена в листинге 4.

Листинг 4 – Реализация клиентского приложения для получения и отображения последних новостей.

import requests  
api\_key = "CG-Z7qPwTTjshcC2vZ4rnpRm1qM"  
response = requests.get(f'https://api.coingecko.com/api/v3/coins/markets?vs\_currency=usd&x\_cg\_api\_key={api\_key}')  
data = response.json()  
str = ""  
  
for coin in data:  
 print(f"{coin['name']} ({coin['symbol']})")  
 print(f"\tprice: {coin['current\_price']}")

Работа программы получения и отображения последних новостей представлена рисунке 9.

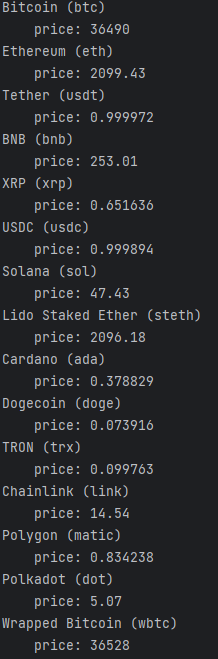


Рисунок 9 – Работа программы получения и отображения последних новостей

1. При выполнении данного задания к ранее созданному API были добавлены аутентификацию и авторизацию пользователей. Реализация данного приложения представлена в листинге 5.

Листинг 5 – Реализация клиентского приложения для получения и отображения последних новостей.

import requests  
from flask import Flask, jsonify  
from flask\_httpauth import HTTPBasicAuth  
app = Flask(\_\_name\_\_)  
auth = HTTPBasicAuth()  
users = {  
 "fles": "zdarova",  
 "denis": "password"  
}  
@auth.verify\_password  
def verify\_password(username, password):  
 if not username or not password:  
 return False  
 if username in users and users[username] == password:  
 return username  
 return False  
  
@app.route('/coins', methods=['GET'])  
@auth.login\_required  
def get\_latest\_news():  
 api\_key = 'CG-Z7qPwTTjshcC2vZ4rnpRm1qM'  
 url = f'https://api.coingecko.com/api/v3/coins/markets?vs\_currency=usd&x\_cg\_api\_key={api\_key}'  
 response = requests.get(url)  
 if response.status\_code == 200:  
 data = response.json()  
 #articles = news\_data['articles']  
 return jsonify({'coins': data}), 200  
 else:  
 return jsonify({'message': 'Ошибка при получении монет'}), 500  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 app.run()

1. **Что такое API?**

API – это набор методов, функций и данных, который позволяет разработчикам взаимодействовать с определенным программным продуктом, приложением или сервисом.

1. **Какую роль играют API в разработке ПО?**

* Абстракция: API скрывает сложность реализации определенной функциональности, предоставляя простой интерфейс для использования. Это позволяет разработчикам сосредотачиваться на более высокоуровневых задачах.
* Совместимость: API определяет стандарты и протоколы, что облегчает интеграцию различных компонентов в одно целое.
* Расширяемость: API может быть расширено и улучшено без изменения самого приложения. Это позволяет сделать приложение более гибким и поддающимся развитию.

1. **Где используются API?**

API используется во множестве областей разработки программного обеспечения:

* веб-сервисы;
* библиотеки и фреймворки;
* операционные системы;
* интернет вещей (IoT).

1. **Какие примеры популярных типов API вы можете назвать?**

Существует несколько распространенных типов API, каждый из которых ориентирован на определенные задачи:

* HTTP API;
* REST API;
* SOAP API.

1. **Почему API важен в разработке программного обеспечения?**

API важны в разработке программного обеспечения, потому что они обеспечивают модульность, переиспользуемость и расширяемость программного кода. Они позволяют разработчикам использовать функциональность, предоставляемую другими программами или сервисами, без необходимости писать весь код с нуля. Это сокращает время разработки, повышает эффективность и упрощает интеграцию различных компонентов программного обеспечения.

1. **Какую роль играют API в интеграции приложений?**

API играет роль моста между разными приложениями и сервисами, позволяя им взаимодействовать, обмениваться данными и совместно решать задачи.

1. **Какие принципы проектирования RESTful API?**

* всё представляется как ресурсы;
* для операция с ресурсами используются HTTP методы;
* ресурсы могут иметь разное представление, такие как JSON, XML, и др;
* не хранят состояние клиента на сервере между запросами;
* позволяет устанавливать связи между ресурсами.

1. **Что такое спецификация API? Какие инструменты используются для её создания?**

Спецификация API - это документация, описывающая структуру и поведение API.

1. **Какие языки программирования могут использоваться для реализации API?**

Для реализации API можно использовать различные языки программирования, такие как Python, Java, JavaScript, Ruby, C#, PHP и другие.

1. **Какие фреймворки широко используются для разработки API?**

Некоторые широко используемые фреймворки для разработки API включают Flask и Django для Python, Node.js для JavaScript, Ruby on Rails для Ruby, Spring для Java и Laravel для PHP.

**Вывод:** в ходе выполнения лабораторной работы были изучены основы работы с API, принципы проектирования RESTful API, интеграция с внешними API, а также вопросы аутентификации и авторизации в API. Эти навыки и знания являются важными для разработки программного обеспечения, особенно при работе с внешними сервисами и создании клиент-серверных приложений.