**Отчёт по лабораторной работе: Разработка веб-сервиса для хранения текстовых заметок**

**1. Цель работы**

Разработать веб-сервис с использованием **FastAPI**, который позволит:

* Создавать, читать, обновлять, удалять текстовые заметки.
* Работать с заметками через **REST API** и **HTTP-методы**.
* Реализовать авторизацию на основе токенов.
* Хранить заметки в отдельных текстовых файлах.

**2. Установка модулей**

Установите необходимые модули с помощью **pip**:

bash

Копировать код

pip install fastapi uvicorn pydantic

**3. Основной код веб-сервиса**

Создайте файл main.py и добавьте следующий код:

python

Копировать код

from fastapi import FastAPI, HTTPException, Depends

from pydantic import BaseModel

from datetime import datetime

import os

import json

import uuid

app = FastAPI()

# Путь к папкам для хранения заметок и токенов

NOTES\_DIR = "notes"

TOKENS\_FILE = "tokens.json"

# Создаём папку для заметок, если её нет

os.makedirs(NOTES\_DIR, exist\_ok=True)

# Модель для текстовой заметки

class Note(BaseModel):

text: str

# Модель ответа о времени создания/обновления

class NoteInfo(BaseModel):

created\_at: str

updated\_at: str

# Модель ответа с текстом заметки

class NoteContent(BaseModel):

id: str

text: str

# Авторизация через токен

def authenticate(token: str):

if not os.path.exists(TOKENS\_FILE):

raise HTTPException(status\_code=401, detail="Unauthorized")

with open(TOKENS\_FILE, "r") as f:

tokens = json.load(f)

if token not in tokens.values():

raise HTTPException(status\_code=401, detail="Invalid token")

return True

# Генерация токенов для тестирования (однократно)

if not os.path.exists(TOKENS\_FILE):

with open(TOKENS\_FILE, "w") as f:

json.dump({"user": "test\_token"}, f)

# API методы

@app.post("/notes/create", response\_model=dict)

def create\_note(note: Note, token: str):

authenticate(token)

note\_id = str(uuid.uuid4())

note\_data = {

"id": note\_id,

"text": note.text,

"created\_at": datetime.utcnow().isoformat(),

"updated\_at": datetime.utcnow().isoformat()

}

with open(os.path.join(NOTES\_DIR, f"{note\_id}.json"), "w") as f:

json.dump(note\_data, f)

return {"id": note\_id}

@app.get("/notes/{note\_id}", response\_model=NoteContent)

def get\_note(note\_id: str, token: str):

authenticate(token)

file\_path = os.path.join(NOTES\_DIR, f"{note\_id}.json")

if not os.path.exists(file\_path):

raise HTTPException(status\_code=404, detail="Note not found")

with open(file\_path, "r") as f:

note = json.load(f)

return NoteContent(id=note["id"], text=note["text"])

@app.get("/notes/info/{note\_id}", response\_model=NoteInfo)

def get\_note\_info(note\_id: str, token: str):

authenticate(token)

file\_path = os.path.join(NOTES\_DIR, f"{note\_id}.json")

if not os.path.exists(file\_path):

raise HTTPException(status\_code=404, detail="Note not found")

with open(file\_path, "r") as f:

note = json.load(f)

return NoteInfo(created\_at=note["created\_at"], updated\_at=note["updated\_at"])

@app.patch("/notes/update/{note\_id}", response\_model=dict)

def update\_note(note\_id: str, note: Note, token: str):

authenticate(token)

file\_path = os.path.join(NOTES\_DIR, f"{note\_id}.json")

if not os.path.exists(file\_path):

raise HTTPException(status\_code=404, detail="Note not found")

with open(file\_path, "r") as f:

note\_data = json.load(f)

note\_data["text"] = note.text

note\_data["updated\_at"] = datetime.utcnow().isoformat()

with open(file\_path, "w") as f:

json.dump(note\_data, f)

return {"message": "Note updated successfully"}

@app.delete("/notes/delete/{note\_id}", response\_model=dict)

def delete\_note(note\_id: str, token: str):

authenticate(token)

file\_path = os.path.join(NOTES\_DIR, f"{note\_id}.json")

if os.path.exists(file\_path):

os.remove(file\_path)

return {"message": "Note deleted successfully"}

raise HTTPException(status\_code=404, detail="Note not found")

@app.get("/notes/list", response\_model=dict)

def list\_notes(token: str):

authenticate(token)

note\_files = os.listdir(NOTES\_DIR)

note\_ids = [note\_file.split(".")[0] for note\_file in note\_files]

return {idx: note\_id for idx, note\_id in enumerate(note\_ids)}

**4. Запуск сервера**

Запустите сервер с помощью команды **uvicorn**:

bash

Копировать код

uvicorn main:app --reload

Сервер будет доступен по адресу:  
http://127.0.0.1:8000

**5. Тестирование API**

1. **Создание заметки (POST):**
   * URL: /notes/create
   * Тело запроса:

json

Копировать код

{"text": "Это моя первая заметка"}

* + Параметр: token=test\_token.

1. **Получение заметки по ID (GET):**
   * URL: /notes/{note\_id}
   * Параметр: token=test\_token.
2. **Получение информации о заметке (GET):**
   * URL: /notes/info/{note\_id}
   * Параметр: token=test\_token.
3. **Обновление заметки (PATCH):**
   * URL: /notes/update/{note\_id}
   * Тело запроса:

json

Копировать код

{"text": "Обновлённая заметка"}

* + Параметр: token=test\_token.

1. **Удаление заметки (DELETE):**
   * URL: /notes/delete/{note\_id}
   * Параметр: token=test\_token.
2. **Получение списка заметок (GET):**
   * URL: /notes/list
   * Параметр: token=test\_token.

**6. Выводы**

В ходе лабораторной работы:

* Разработан веб-сервис для хранения текстовых заметок на **FastAPI**.
* Реализованы методы API: создание, получение, обновление, удаление заметок и получение списка.
* Добавлен механизм авторизации на основе токенов.
* Все данные хранятся в отдельных JSON-файлах.