Ministerul Educației și Cercetării al Republicii Moldova

Universitatea Tehnică a Moldovei

Facultatea Calculatoare, Informatică și Microelectronică

Departamentul Ingineria Software și Automatică

**Raport**

Disciplina: *Programarea în rețea*  
Lucrarea de laborator nr.3  
Tema: Aplicație de tip client DNS

A efectuat: st.gr. TI-225, Rotari Cristian

A verificat: lect.univ. Ion Gatman

Chișinău 2025

Cod:

import socket

import dns.resolver

import dns.reversename

import dns.exception

import ipaddress

current\_dns = None

*def* **is\_valid\_ip**(ip\_str):

    try:

        ipaddress.*ip\_address*(ip\_str)

        return True

    except ValueError:

        return False

*def* **validate\_dns\_server**(dns\_ip):

    resolver = dns.resolver.*Resolver*()

    resolver.nameservers = [dns\_ip]

    resolver.lifetime = 2

    try:

        resolver.*resolve*('example.com','A')

        return True

    except dns.exception.DNSException:

        return False

*def* **resolve\_domain\_system**(domain):

    try:

        \_, \_, ips = socket.*gethostbyname\_ex*(domain)

        return ips, []

    except socket.gaierror as e:

        return None, [*str*(e)]

*def* **resolve\_ip\_system**(ip):

    try:

        hostname, \_, \_ = socket.*gethostbyaddr*(ip)

        return [hostname], []

    except socket.herror as e:

        return None, [*str*(e)]

*def* **resolve\_domain\_custom**(domain, dns\_ip):

    resolver = dns.resolver.*Resolver*()

    resolver.nameservers = [dns\_ip]

    resolver.lifetime = 2

    ips = []

    errors = []

    try:

        answers\_a = resolver.*resolve*(domain,'A')

        ips.*extend*([*str*(r)forrinanswers\_a])

    except dns.resolver.NXDOMAIN:

        return None, ["Domain not found."]

    except dns.resolver.NoAnswer:

        pass

    except dns.resolver.Timeout:

        errors.*append*("A record query timed out.")

    except dns.exception.DNSException as e:

        errors.*append*(*f*"A record error: *{*e*}*")

    try:

        answers\_aaaa = resolver.*resolve*(domain,'AAAA')

        ips.*extend*([*str*(r)forrinanswers\_aaaa])

    except dns.resolver.NXDOMAIN:

        errors.*append*("Domain not found during AAAA query.")

    except dns.resolver.NoAnswer:

        pass

    except dns.resolver.Timeout:

        errors.*append*("AAAA record query timed out.")

    except dns.exception.DNSException as e:

        errors.*append*(*f*"AAAA record error: *{*e*}*")

    if not ips and errors:

        return None, errors

    elif not ips and not errors:

        return None, ["No IP addresses found."]

    return ips, errors

*def* **resolve\_ip\_custom**(ip\_str, dns\_ip):

    resolver = dns.resolver.*Resolver*()

    resolver.nameservers = [dns\_ip]

    resolver.lifetime = 2

    try:

        rev\_name = dns.reversename.*from\_address*(ip\_str)

        answers = resolver.*resolve*(rev\_name,'PTR')

        domains = [*str*(r.*target*) for r in answers]

        return domains, []

    except dns.resolver.NXDOMAIN:

        return None, ["No PTR records found."]

    except dns.resolver.NoAnswer:

        return None, ["No PTR records found."]

    except dns.resolver.Timeout:

        return None, ["DNS query timed out."]

    except dns.exception.DNSException as e:

        return None, [*f*"DNS error: {e}"]

*def* **main**():

*global* current\_dns

*print*("DNS Client Application. Commands: 'resolve <domain/ip>' or 'use dns <ip>'")

    while True:

        try:

            command = *input*("> ").*strip*()

            if not command:

                continue

            parts = command.*split*()

            if not parts:

                continue

            if parts[0] == "resolve" and *len*(parts) == 2:

                target = parts[1]

                is\_ip = False

                try:

                    ipaddress.*ip\_address*(target)

                    is\_ip = True

                except ValueError:

                    pass

                if current\_dns is None:

                    if is\_ip:

                        result, errors = *resolve\_ip\_system*(target)

                    else:

                        result, errors = *resolve\_domain\_system*(target)

                else:

                    if is\_ip:

                        result, errors = *resolve\_ip\_custom*(target,current\_dns)

                    else:

                        result, errors = *resolve\_domain\_custom*(target,current\_dns)

                if errors:

*print*("Errors:")

                    for error in errors:

*print*(*f*" - *{*error*}*")

                if result:

*print*("Domains:"ifis\_ipelse"IPs:")

                    for item in result:

*print*(item)

                elif not errors:

*print*("No results found.")

            elif parts[0] == "use" and *len*(parts) == 3 and parts[1] == "dns":

                new\_dns = parts[2]

                if not *is\_valid\_ip*(new\_dns):

*print*("Error: Invalid IP address format")

                    continue

                if *validate\_dns\_server*(new\_dns):

                    current\_dns = new\_dns

*print*(*f*"DNS server changed to *{*new\_dns*}*")

                else:

*print*("Error: DNS server is unreachable or not responding")

            else:

*print*("Invalid command. Usage:")

*print*("resolve <domain/ip>")

*print*("use dns <ip>")

        except KeyboardInterrupt:

*print*("\nExiting...")

            break

        except Exception as e:

*print*(*f*"Error: *{str*(e)*}*")

if \_\_name\_\_ == "\_\_main\_\_":

*main*()

Rezultatul:

