Практическая работа 5
Сетевой терминал

```
import flet as ft
import random
import time
import threading
from datetime import datetime

class Packet:
    def __init__(self, packet_id, source, destination, size):
        self.id = packet_id
        self.source = source
        self.destination = destination
        self.size = size
        self.start_time = time.time()
        self.at_switch = False
        self.progress = 0
        self.control = ft.Container()

class NetworkTerminal:
    def __init__(self, page: ft.Page):
        self.page = page
        self.page.title = "Сетевой терминал"
        self.page.window.width = 1000
        self.page.window.height = 700

        self.running = False
        self.packet_counter = 0
        self.packets = []
        self.packets_sent = 0
        self.packets_delivered = 0
        self.total_delay = 0

        self.pcs = {
            "ПК1": {"x": 50, "y": 50},
            "ПК2": {"x": 350, "y": 50},
            "ПК3": {"x": 50, "y": 350},
            "ПК4": {"x": 350, "y": 350}
        }

        self.switch_x = 200
```

```
        self.switch_y = 200
        self.speed = 3

        self.packet_controls = []

        self.setup_ui()

    def setup_ui(self):
        self.console = ft.TextField(
            multiline=True,
            read_only=True,
            width=380,
            height=450,
            bgcolor="black",
            color="green",
            text_style=ft.TextStyle(font_family="Courier")
        )

        self.speed_slider = ft.Slider(
            min=1,
            max=10,
            divisions=9,
            label="Скорость: {value}",
            value=3,
            on_change=self.on_speed_change
        )

        self.stats_text = ft.Text("Пакетов отправлено: 0\nПакетов доставлено: 0",
size=14)

        self.network_area = ft.Stack(
            width=450,
            height=450,
            controls=self.create_network_controls()
        )

        start_btn = ft.ElevatedButton("Старт", on_click=self.start_simulation,
bgcolor="green")
        stop_btn = ft.ElevatedButton("Стоп", on_click=self.stop_simulation,
bgcolor="red")
        clear_btn = ft.ElevatedButton("Очистить", on_click=self.clear_console)

        left_panel = ft.Container(
```

```
            content=ft.Column([
                ft.Text("Сетевой терминал", size=20, weight=ft.FontWeight.BOLD),
                self.network_area,
                ft.Row([start_btn, stop_btn, clear_btn]),
                ft.Text("Скорость передачи (пакетов/сек):"),
                self.speed_slider,
                self.stats_text
            ]),
            padding=10
        )

        right_panel = ft.Container(
            content=ft.Column([
                ft.Text("Консоль логов", size=20, weight=ft.FontWeight.BOLD),
                self.console
            ]),
            padding=10
        )

        self.page.add(ft.Row([left_panel, right_panel]))

    def create_network_controls(self):
        controls = []

        for pc_name, pc_data in self.pcs.items():
            pc = ft.Container(
                content=ft.Column([
                    ft.Container(
                        content=ft.Icon(ft.Icons.COMPUTER, size=30, color="white"),
                        bgcolor="grey",
                        width=60,
                        height=40,
                        border_radius=5,
                        alignment=ft.alignment.center
                    ),
                    ft.Text(pc_name, size=10, color="white")
                ]),
                left=pc_data["x"],
                top=pc_data["y"],
                width=70
            )
            controls.append(pc)
```

```
        switch = ft.Container(
            content=ft.Column([
                ft.Container(
                    content=ft.Icon(ft.Icons.ROUTER, size=40, color="black"),
                    bgcolor="cyan",
                    width=80,
                    height=50,
                    border_radius=5,
                    alignment=ft.alignment.center
                ),
                ft.Text("SWITCH", size=10, color="white", weight=ft.FontWeight.BOLD)
            ]),
            left=self.switch_x - 40,
            top=self.switch_y - 25,
            width=90
        )
        controls.append(switch)

        return controls

    def log(self, message):
        timestamp = datetime.now().strftime("[%H:%M:%S.%f")[:-3] + "]"
        if self.console.value is None:
            self.console.value = ""
        self.console.value += f"{timestamp} {message}\n"
        self.console.update()

    def on_speed_change(self, e):
        self.speed = int(e.control.value)

    def start_simulation(self, e):
        if not self.running:
            self.running = True
            threading.Thread(target=self.simulation_loop, daemon=True).start()

    def stop_simulation(self, e):
        self.running = False
        self.log(f"=== ИТОГОВАЯ СТАТИСТИКА ===")
        self.log(f"Всего передано пакетов: {self.packets_sent}")
        self.log(f"Доставлено пакетов: {self.packets_delivered}")
        if self.packets_delivered > 0:
            avg_delay = self.total_delay / self.packets_delivered
            self.log(f"Средняя задержка: {avg_delay:.0f} мс")
```

```
    def clear_console(self, e):
        self.console.value = ""
        self.console.update()

    def simulation_loop(self):
        while self.running:
            self.create_packet()
            self.update_packets()
            time.sleep(1 / self.speed)

    def create_packet(self):
        self.packet_counter += 1
        self.packets_sent += 1

        sources = list(self.pcs.keys())
        source = random.choice(sources)
        destinations = [pc for pc in sources if pc != source]
        destination = random.choice(destinations)

        size = random.randint(100, 500)

        packet = Packet(self.packet_counter, source, destination, size)

        src_data = self.pcs[source]
        packet.control = ft.Container(
            content=ft.Text(f"#{packet.id}", size=8, color="black",
weight=ft.FontWeight.BOLD),
            bgcolor="yellow",
            width=25,
            height=20,
            border_radius=10,
            alignment=ft.alignment.center,
            left=src_data["x"] + 25,
            top=src_data["y"] + 20
        )

        self.packets.append(packet)
        self.network_area.controls.append(packet.control)
        self.network_area.update()

        self.log(f"Пакет #{packet.id}: {source} -> {destination}, Размер: {size} байт")
        self.update_stats()
```

```
    def update_packets(self):
        packets_to_remove = []

        for packet in self.packets:
            src_data = self.pcs[packet.source]
            dst_data = self.pcs[packet.destination]

            src_x = src_data["x"] + 25
            src_y = src_data["y"] + 20
            dst_x = dst_data["x"] + 25
            dst_y = dst_data["y"] + 20

            packet.progress += 0.1

            if not packet.at_switch:
                packet.control.left = src_x + (self.switch_x - src_x) * packet.progress
                packet.control.top = src_y + (self.switch_y - src_y) * packet.progress

                if packet.progress >= 1.0:
                    packet.at_switch = True
                    packet.progress = 0
                    self.log(f"Пакет #{packet.id} достиг SWITCH")
            else:
                packet.control.left = self.switch_x + (dst_x - self.switch_x) *
packet.progress
                packet.control.top = self.switch_y + (dst_y - self.switch_y) *
packet.progress

                if packet.progress >= 1.0:
                    delay = (time.time() - packet.start_time) * 1000
                    self.packets_delivered += 1
                    self.total_delay += delay
                    self.log(f"Пакет #{packet.id} доставлен на {packet.destination}
(задержка: {delay:.0f} мс)")
                    packets_to_remove.append(packet)
                    self.update_stats()

        for packet in packets_to_remove:
            self.packets.remove(packet)
            if packet.control in self.network_area.controls:
                self.network_area.controls.remove(packet.control)
```

```
        self.network_area.update()

    def update_stats(self):
        self.stats_text.value = f"Пакетов отправлено: {self.packets_sent}\nПакетов доставлено: {self.packets_delivered}"
        self.stats_text.update()

def main(page: ft.Page):
    NetworkTerminal(page)

ft.app(target=main)
```

Теперь посмотрим на работу приложения на рисунке 1:

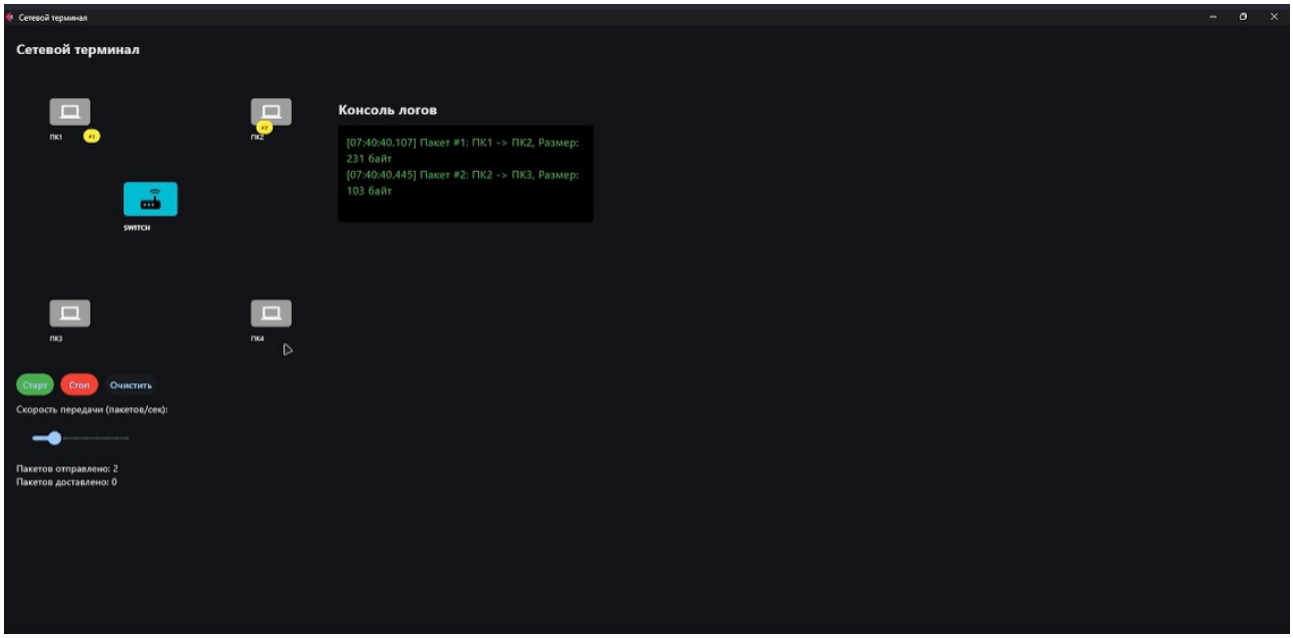

Рисунок 1 — Демонстрация работы