

Практическая работа 7

Разработка приложения для построения схем метро

Для начала посмотрим код:

```
import flet as ft
import matplotlib.pyplot as plt
from matplotlib.backends.backend_agg import FigureCanvasAgg
import io
import base64
import numpy as np

def main(page: ft.Page):
    page.title = "Анализатор механического движения"
    page.window_width = 800
    page.window_height = 700

    def calculate(e):
        try:
            v0 = float(initial_velocity.value)
            a = float(acceleration.value)
            t = float(time.value)

            if t < 0:
                result_text.value = "Ошибка: время не может быть отрицательным"
                page.update()
                return

            if a > 0:
                movement_type = "Равноускоренное движение"
                description = f"Тело разгоняется с ускорением {a} м/с²"
            elif a < 0:
                movement_type = "Равнозамедленное движение"
                description = f"Тело замедляется с ускорением {abs(a)} м/с²"
            else:
                movement_type = "Равномерное движение"
                description = "Тело движется с постоянной скоростью"

            s = v0 * t + 0.5 * a * t ** 2
            v_final = v0 + a * t

            if s < 0:
                s = abs(s)

            type_result.value = movement_type
```

```

distance_result.value = f"{s:.2f} м"
final_velocity_result.value = f"{v_final:.2f} м/с"
description_result.value = description

if a == 0:
    equation_text.value = f"S(t) = {v0} · t"
else:
    equation_text.value = f"S(t) = {v0} · t + {0.5 * a:.2f} · t2"

times = np.linspace(0, t, 100)
distances = v0 * times + 0.5 * a * times ** 2

fig, ax = plt.subplots(figsize=(6, 4))
ax.plot(times, distances, 'b-', linewidth=2, label='S(t)')

key_times = np.linspace(0, t, 10)
key_distances = v0 * key_times + 0.5 * a * key_times ** 2
ax.scatter(key_times, key_distances, color='red', s=30, zorder=5)

ax.set_xlabel('Время t (с)')
ax.set_ylabel('Путь S (м)')
ax.set_title('График зависимости пути от времени')
ax.grid(True)
ax.legend()

buf = io.BytesIO()
plt.savefig(buf, format='png', dpi=100, bbox_inches='tight')
buf.seek(0)
plt.close()

img_base64 = base64.b64encode(buf.read()).decode('utf-8')
chart_img.src_base64 = img_base64

page.update()

except ValueError:
    result_text.value = "Ошибка: введите корректные числовые значения"
    page.update()

def clear(e):
    initial_velocity.value = ""
    acceleration.value = ""
    time.value = ""
    type_result.value = ""
    distance_result.value = ""

```

```
final_velocity_result.value = ""
description_result.value = ""
equation_text.value = ""
chart_img.src_base64 = ""
result_text.value = ""
page.update()
```

```
initial_velocity = ft.TextField(label="Начальная скорость  $v_0$  (м/с)", width=300)
acceleration = ft.TextField(label="Ускорение  $a$  (м/с2)", width=300)
time = ft.TextField(label="Время движения  $t$  (с)", width=300)
```

```
result_text = ft.Text("", color=ft.Colors.RED)
```

```
type_result = ft.Text("", size=16)
distance_result = ft.Text("", size=16)
final_velocity_result = ft.Text("", size=16)
description_result = ft.Text("", size=14)
```

```
equation_text = ft.Text("", size=16, weight=ft.FontWeight.BOLD)
```

```
chart_img = ft.Image(width=600, height=400)
```

```
input_section = ft.Container(
    content=ft.Column([
        ft.Text("Входные данные", size=20, weight=ft.FontWeight.BOLD),
        initial_velocity,
        acceleration,
        time,
        ft.Row([
            ft.ElevatedButton("Рассчитать", on_click=calculate),
            ft.ElevatedButton("Очистить", on_click=clear)
        ])
    ]),
    padding=10
)
```

```
results_section = ft.Container(
    content=ft.Column([
        ft.Text("Результаты", size=20, weight=ft.FontWeight.BOLD),
        ft.Row([ft.Text("Тип движения:", weight=ft.FontWeight.BOLD),
type_result]),
        ft.Row([ft.Text("Пройденный путь:", weight=ft.FontWeight.BOLD),
distance_result]),
        ft.Row([ft.Text("Конечная скорость:", weight=ft.FontWeight.BOLD),
final_velocity_result])
    ])
)
```

```

        ft.Row([ft.Text("Описание:", weight=ft.FontWeight.BOLD),
description_result]),
        result_text
    ]),
    padding=10
)

chart_section = ft.Container(
    content=ft.Column([
        ft.Text("График", size=20, weight=ft.FontWeight.BOLD),
        chart_img,
        equation_text
    ]),
    padding=10
)

page.add(
    ft.Row([
        ft.Container(content=input_section, width=350),
        ft.Container(content=results_section, width=400)
    ]),
    ft.Divider(),
    chart_section
)

ft.app(target=main)

```

Теперь посмотрим на работу приложения на рисунке 1:

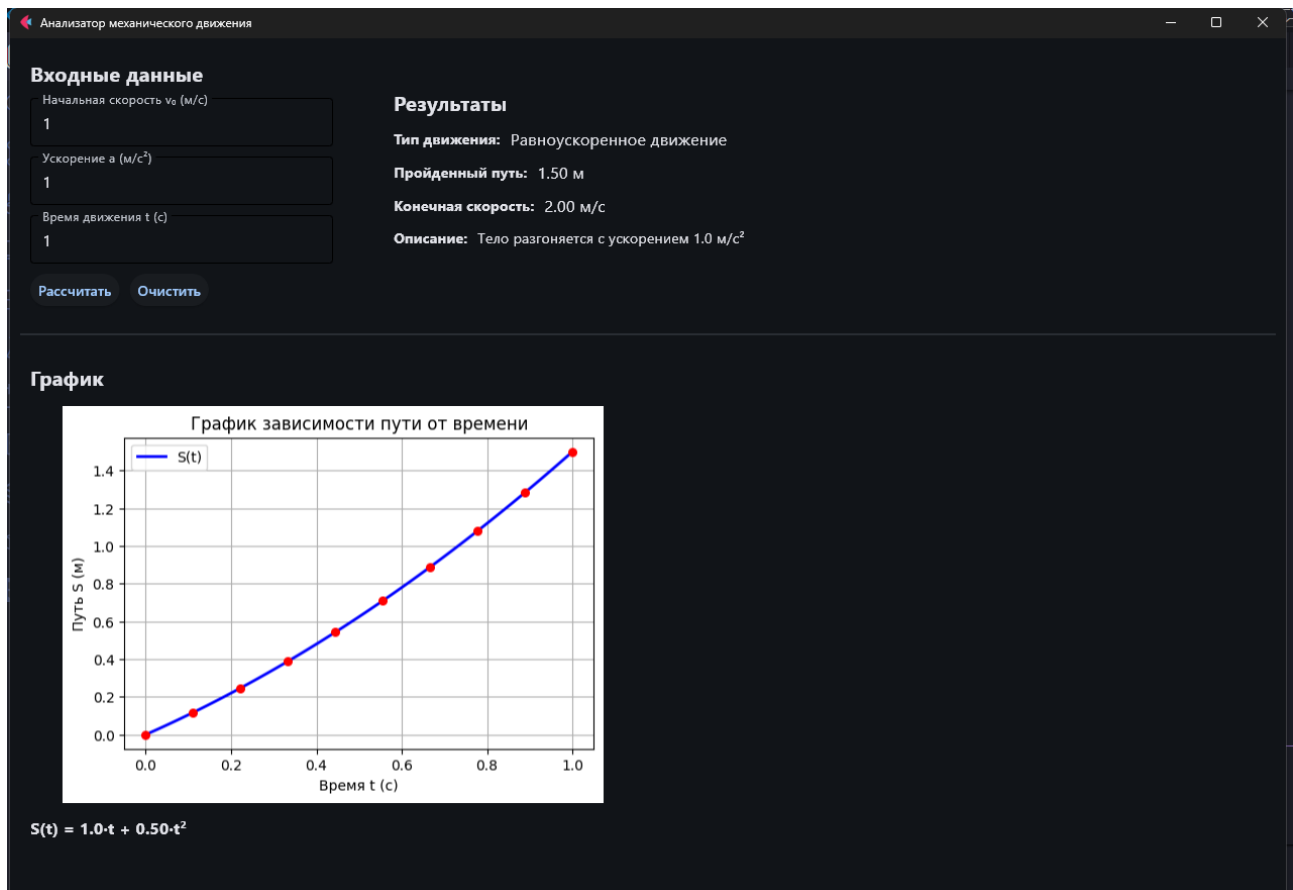


Рисунок 1 — Демонстрация работы