Практическая работа 11
Создание чего-то с графиком

Для начала рассмотрим код

```
import flet as ft
import matplotlib.pyplot as plt
from scipy.stats import norm
import numpy as np
import sqlite3
from datetime import datetime
import openpyxl
import io
import base64
from reportlab.lib.pagesizes import A4
from reportlab.lib.styles import getSampleStyleSheet
from reportlab.platypus import SimpleDocTemplate, Paragraph, Spacer, Image
from reportlab.lib.enums import TA_CENTER
from reportlab.pdfbase import pdfmetrics
from reportlab.pdfbase.ttfonts import TTFont

def init_db():
    conn = sqlite3.connect('quality_indexes.db')
    cursor = conn.cursor()
    cursor.execute('''CREATE TABLE IF NOT EXISTS calculations
                   (id INTEGER PRIMARY KEY AUTOINCREMENT,
                    usl REAL,
                    lsl REAL,
                    mean REAL,
                    std REAL,
                    cp REAL,
                    cpk REAL,
                    status TEXT,
                    timestamp TEXT)''')
    conn.commit()
    conn.close()

def save_calculation(usl, lsl, mean, std, cp, cpk, status):
    conn = sqlite3.connect('quality_indexes.db')
    cursor = conn.cursor()
    cursor.execute('INSERT INTO calculations (usl, lsl, mean, std, cp, cpk, status,
timestamp) VALUES (?, ?, ?, ?, ?, ?, ?, ?)',
                   (usl, lsl, mean, std, cp, cpk, status, datetime.now().strftime('%Y-%m-%d
%H:%M:%S')))
    conn.commit()
```

```
    conn.close()

def load_history():
    conn = sqlite3.connect('quality_indexes.db')
    cursor = conn.cursor()
    cursor.execute('SELECT id, usl, lsl, mean, std, cp, cpk, status, timestamp FROM
calculations ORDER BY id DESC')
    rows = cursor.fetchall()
    conn.close()
    return rows

def clear_history():
    conn = sqlite3.connect('quality_indexes.db')
    cursor = conn.cursor()
    cursor.execute('DELETE FROM calculations')
    conn.commit()
    conn.close()

def export_to_excel(filename, data):
    wb = openpyxl.Workbook()
    ws = wb.active
    ws.title = "История расчетов"

    headers = ["ID", "Верхняя граница", "Нижняя граница", "Среднее",
"Стандартное отклонение", "Cp", "Cpk", "Статус", "Время"]
    for i, header in enumerate(headers):
        ws.cell(row=1, column=i+1, value=header)

    for i, row in enumerate(data):
        for j, value in enumerate(row):
            ws.cell(row=i+2, column=j+1, value=value)

    wb.save(filename)

def import_from_excel(filename):
    try:
        wb = openpyxl.load_workbook(filename)
        ws = wb.active
        data = []
        for row in ws.iter_rows(min_row=2):
            if row[0].value:
                data.append([cell.value for cell in row])
        return data
    except:
        return []
```

```
def export_to_pdf(filename, usl, lsl, mean, std, cp, cpk, status):
    try:
        pdfmetrics.registerFont(TTFont('Arial', 'arial.ttf'))
        pdfmetrics.registerFont(TTFont('Arial-Bold', 'arialbd.ttf'))
    except:
        pass

    doc = SimpleDocTemplate(filename, pagesize=A4)
    styles = getSampleStyleSheet()

    style = styles['Normal']
    style.fontName = 'Arial'
    style.leading = 14

    style_title = styles['Title']
    style_title.fontName = 'Arial-Bold'
    style_title.alignment = TA_CENTER

    style_heading = styles['Heading2']
    style_heading.fontName = 'Arial-Bold'

    elements = []

    title = Paragraph("Анализ индексов качества процесса", style_title)
    elements.append(title)
    elements.append(Spacer(1, 20))

    elements.append(Paragraph("Результаты расчета", style_heading))
    elements.append(Spacer(1, 12))

    text = f"""Верхняя граница допуска (USL): {usl:.2f}
Нижняя граница допуска (LSL): {lsl:.2f}
Среднее арифметическое (μ): {mean:.2f}
Стандартное отклонение (σ): {std:.2f}
Индекс воспроизводимости (Cp): {cp:.3f}
Индекс пригодности (Cpk): {cpk:.3f}
Статус: {status}"""

    elements.append(Paragraph(text, style))
    elements.append(Spacer(1, 20))

    doc.build(elements)

class QualityAnalyzer:
```

```
    def __init__(self, page: ft.Page):
        self.page = page
        self.page.title = "Анализ индексов качества процесса"
        self.page.window.width = 1200
        self.page.window.height = 700

        init_db()

        self.file_picker = ft.FilePicker(on_result=self.pick_files_result)
        self.page.overlay.append(self.file_picker)

        self.setup_ui()

    def setup_ui(self):
        self.usl_input = ft.TextField(label="Верхняя граница допуска (USL)",
width=200)
        self.lsl_input = ft.TextField(label="Нижняя граница допуска (LSL)",
width=200)
        self.mean_input = ft.TextField(label="Среднее арифметическое (μ)",
width=200)
        self.std_input = ft.TextField(label="Стандартное отклонение (σ)", width=200)

        self.cp_result = ft.Text("", size=14, weight=ft.FontWeight.BOLD)
        self.cpk_result = ft.Text("", size=14, weight=ft.FontWeight.BOLD)
        self.status_result = ft.Text("", size=14, weight=ft.FontWeight.BOLD)
        self.action_result = ft.Text("", size=12, color="blue")

        self.calculate_btn = ft.ElevatedButton("Рассчитать", on_click=self.calculate)
        self.save_btn = ft.ElevatedButton("Сохранить в историю",
on_click=self.save_result, visible=False)
        self.export_excel_btn = ft.ElevatedButton("Экспорт в Excel",
on_click=self.export_current, visible=False)
        self.export_pdf_btn = ft.ElevatedButton("Экспорт в PDF",
on_click=self.export_to_pdf_current, visible=False)
        self.import_excel_btn = ft.ElevatedButton("Импорт из Excel",
on_click=self.import_excel)
        self.open_chart_btn = ft.ElevatedButton("Открыть график",
on_click=self.open_chart_window, visible=False)

        calculator_content = ft.Container(
            content=ft.Column([
                ft.Row([self.usl_input, self.lsl_input],
alignment=ft.MainAxisAlignment.CENTER),
                ft.Row([self.mean_input, self.std_input],
alignment=ft.MainAxisAlignment.CENTER),
```

```
            ft.Container(height=20),
            ft.Row([self.calculate_btn, self.import_excel_btn],
alignment=ft.MainAxisAlignment.CENTER),
            ft.Container(height=20),
            ft.Row([self.save_btn, self.export_excel_btn, self.export_pdf_btn,
self.open_chart_btn], alignment=ft.MainAxisAlignment.CENTER),
            ft.Container(height=20),
            ft.Text("Результаты расчета:", size=16, weight=ft.FontWeight.BOLD),
            ft.Container(height=5),
            self.cp_result,
            self.cpk_result,
            self.status_result,
            self.action_result
        ]),
        padding=20
    )

    self.chart_image = ft.Image(width=700, height=400)
    refresh_chart_btn = ft.ElevatedButton("Обновить график",
on_click=self.refresh_chart)
    chart_content = ft.Container(
        content=ft.Column([
            ft.Text("График распределения", size=18, weight=ft.FontWeight.BOLD),
            ft.Container(height=10),
            self.chart_image,
            ft.Container(height=10),
            refresh_chart_btn,
            ft.Container(height=10),
            ft.Text("Красная зона: вне допуска\nЗеленая зона: в допуске", size=12,
color="grey")
        ], alignment=ft.MainAxisAlignment.CENTER),
        padding=20
    )
```

И теперь посмотрим функционал как на рисунке 1:

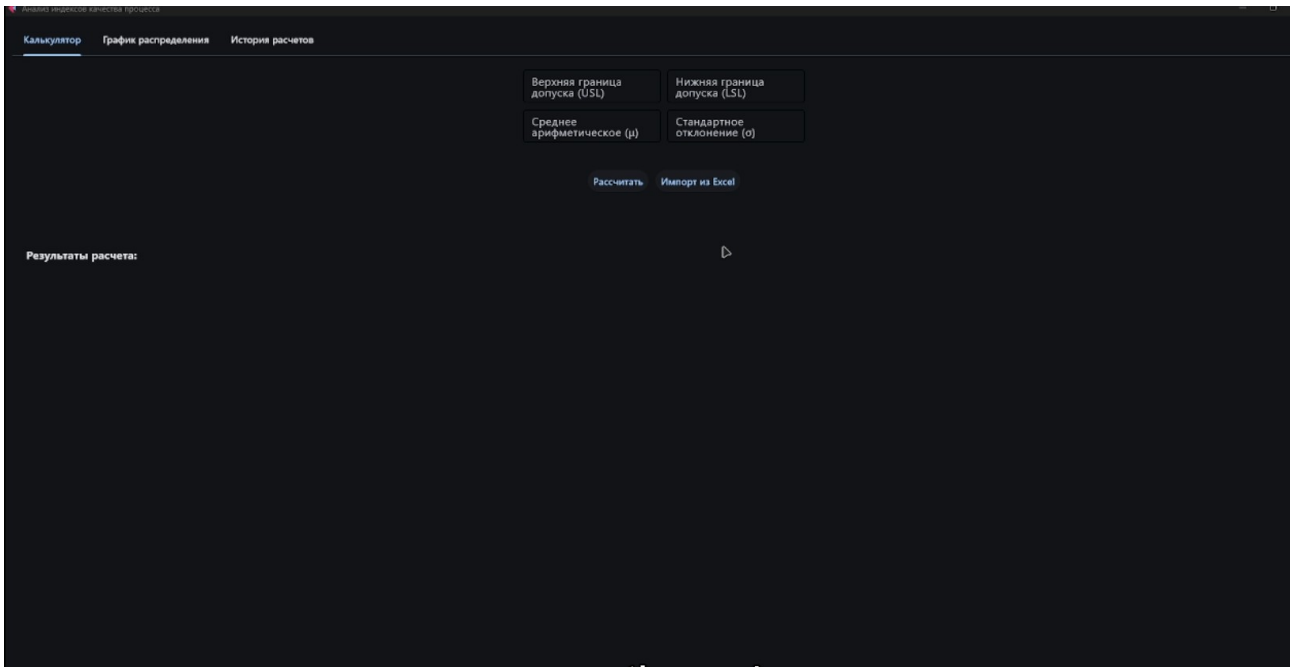

Рисунок 1 — Демонстрация работы