ВЫЧИСЛЕНИЕ ОСОБЫХ ТОЧЕК И ИХ ТИПОВ ДИФФЕРЕНЦИАЛЬНОГО УРАВНЕНИЯ

Буриева Шахзода Акмаловна НММбд-01-23

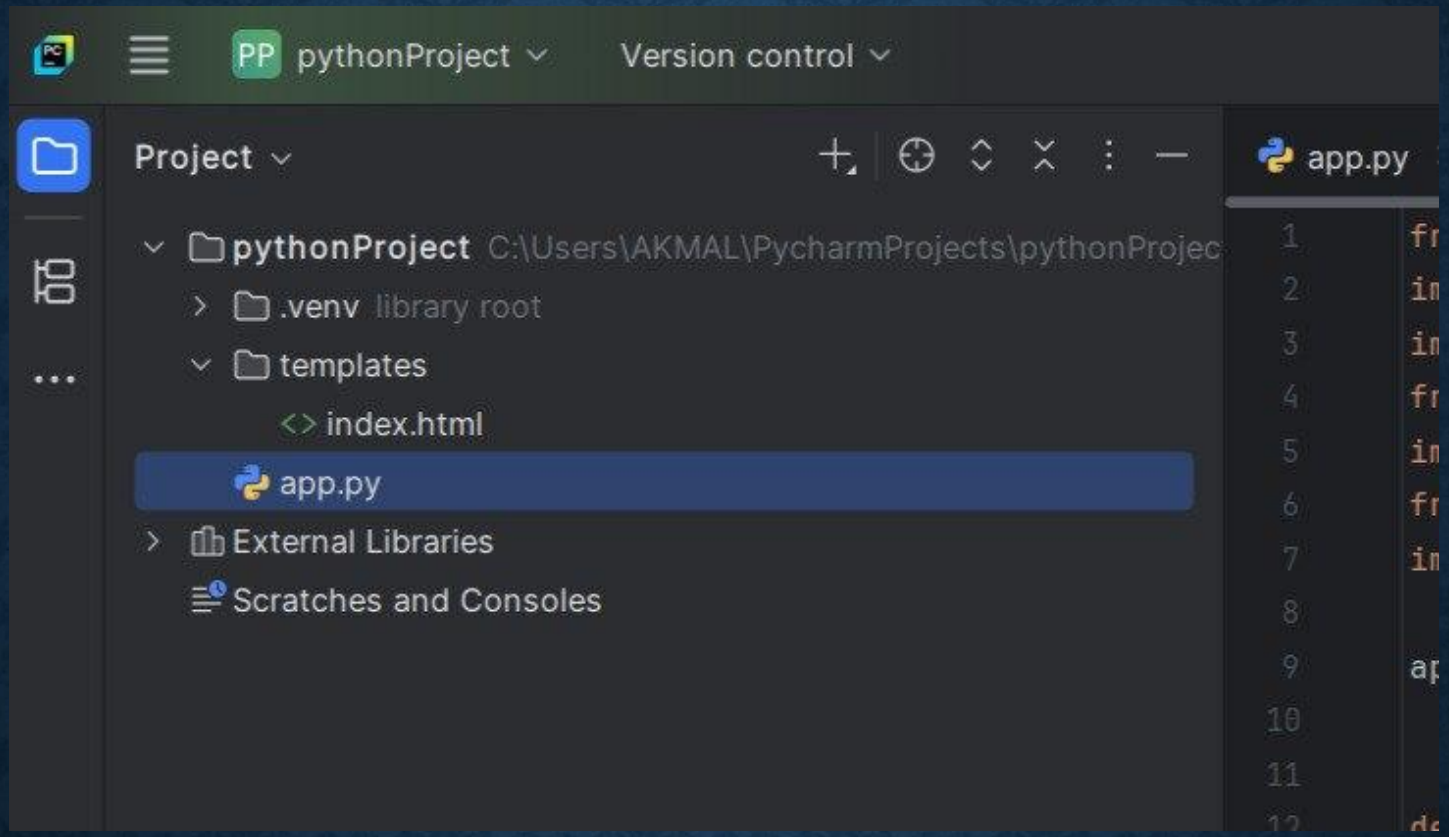
Быкова Алина Александровна НММбд-01-23

АКТУАЛЬНОСТЬ

- 1. **Образовательные цели:** Многие студенты сталкиваются с трудностями в изучении дифференциальных уравнений. Интерактивный сайт может помочь в понимании теории, предоставляя пользователям возможность визуализировать и вычислять особые точки и их типы.
- 2. **Автоматизация вычислений:** Ручные расчеты могут быть трудоемкими и подвержены ошибкам. Сайт, который автоматически вычисляет особые точки и их типы, может значительно упростить процесс и повысить точность результатов.
- 3. **Доступность информации:** Такой ресурс может стать центральным местом для получения информации о методах решения дифференциальных уравнений, что особенно важно для студентов, аспирантов и исследователей.

```
from flask import Flask, render_template, request, jsonify
import numpy as np
import sympy as sp
from sympy.abc import x, y
import matplotlib.pyplot as plt
from io import BytesIO
import base64
```

КАКИЕ ТЕХНОЛОГИИ МЫ ИСПОЛЬЗОВАЛИ?



АРХИТЕКТУРА

ПОКАЗ ПРОЕКТА

```python
def analyze_critical_points(f_expression, g_expression):
    """Анализирует особые точки системы дифференциальных уравнений"""
    try:
        f = sp.sympify(f_expression)
        g = sp.sympify(g_expression)

        # Находим особые точки (где f=0 и g=0)
        critical_points = sp.solve([f, g], (x, y), dict=True)

        results = []
```

```python
def classify_critical_point(eigenvalues):
    """Классифицирует особую точку по собственным значениям"""
    evals = list(eigenvalues.keys())

    # Проверяем количество собственных значений
    if len(evals) < 1:
        return "Нет собственных значений"

    lambda1 = eigenvalues[evals[0]]
    lambda2 = eigenvalues[evals[1]] if len(evals) > 1 else lambda1
```

```python
def plot_phase_portrait(f_expr, g_expr, points, x_range=(-5, 5), y_range=(-5, 5)):
    """Строит фазовый портрет"""
    try:
        # Преобразуем выражения в функции
        f_func = sp.lambdify((x, y), f_expr, 'numpy')
        g_func = sp.lambdify((x, y), g_expr, 'numpy')

        # Создаем сетку
        x_vals = np.linspace(x_range[0], x_range[1], 20)
        y_vals = np.linspace(y_range[0], y_range[1], 20)
        X, Y = np.meshgrid(x_vals, y_vals)

        # Вычисляем производные
        U = f_func(X, Y)
        V = g_func(X, Y)
```

СПАСИБО ЗА ВНИМАНИЕ!