

## МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

# «МИРЭА – Российский технологический университет»

**РТУ МИРЭА**

## Институт комплексной безопасности и специального приборостроения Кафедра КБ-9 «Предметно-ориентированные информационные системы»

Практическая работа №13

# по дисциплине

## «Организация аналитической работы»

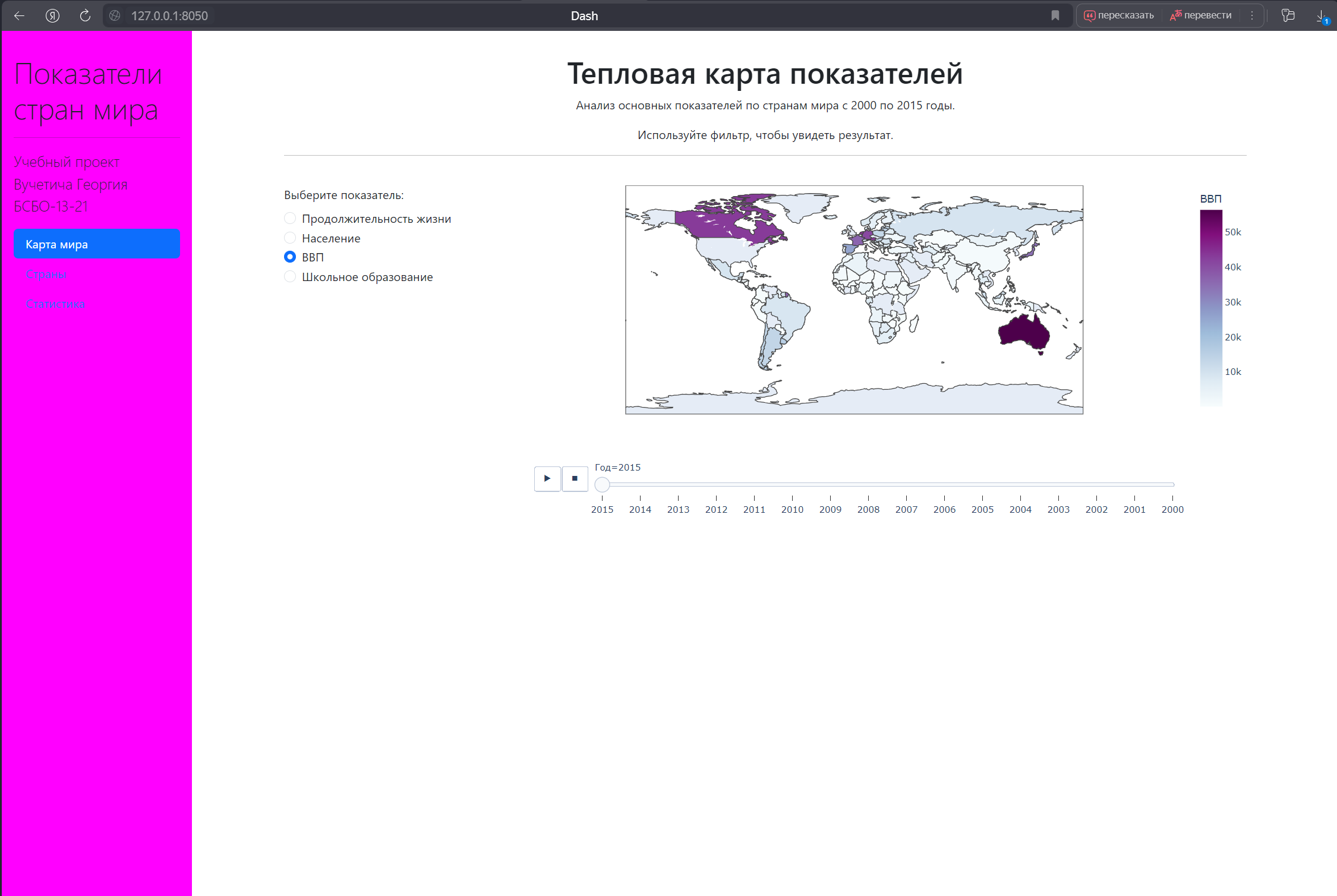
наименование дисциплины

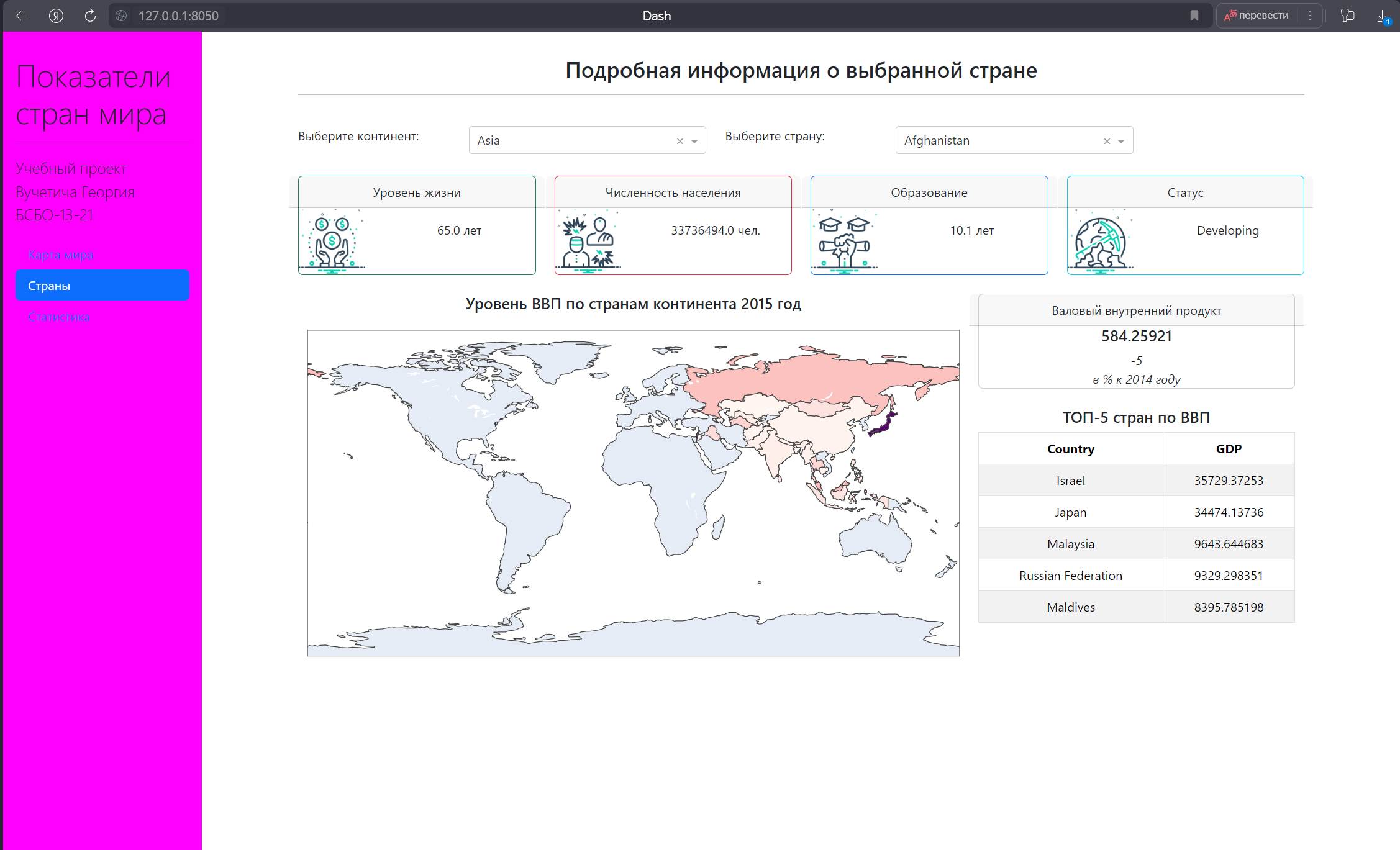
## Студент группы БСБО-13-21 Вучетич Г. В.

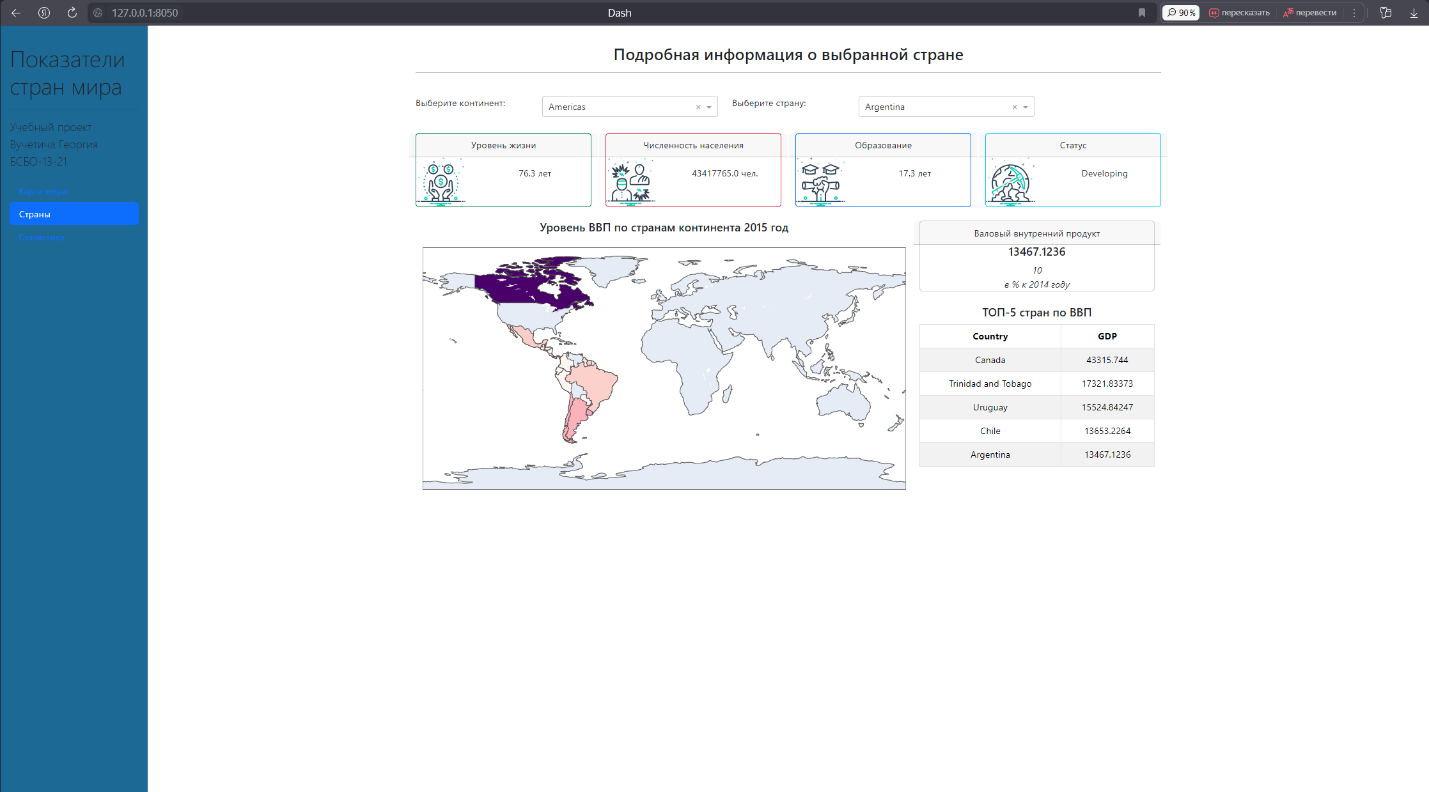
(учебная группа) Фамилия И. О.

## Москва, 2024 г.

# Скриншоты дашбордов:







1. Листинг кодов:

App3.py

import dash

import dash\_bootstrap\_components as dbc

from dash import Input, Output, dcc, html

from dash import Dash, Input, Output, dcc, html

from Pages import all\_map, country, indicators

external\_stylesheets = [dbc.themes.Cyborg]  # Вместо FLATLY выберите свою тему из https://bootswatch.com/

app = Dash(\_\_name\_\_, external\_stylesheets=external\_stylesheets,  use\_pages=True)

app.config.suppress\_callback\_exceptions = True

app = dash.Dash(external\_stylesheets=[dbc.themes.BOOTSTRAP])

## Задаем аргументы стиля для боковой панели. Мы используем position:fixed и фиксированную ширину

SIDEBAR\_STYLE = {

    "position": "fixed",

    "top": 0,

    "left": 0,

    "bottom": 0,

    "width": "16rem",

    "padding": "2rem 1rem",

    "background-color": "#FF00FF", # Цвет фона боковой панели меняем на тот, который больше всего подходит

}

# Справа от боковой панели размешается основной дашборд. Добавим отступы

CONTENT\_STYLE = {

    "margin-left": "18rem",

    "margin-right": "2rem",

    "padding": "2rem 1rem",

}

sidebar = html.Div(

    [

        html.H2("Показатели стран мира", className="display-6"),

        html.Hr(),

        html.P(

            "Учебный проект Вучетича Георгия БСБО-13-21", className="lead"

        ),

        dbc.Nav(

            [

                dbc.NavLink("Карта мира", href="/", active="exact"),

                dbc.NavLink("Страны", href="/page-1", active="exact"),

                dbc.NavLink("Статистика", href="/page-2", active="exact"),

            ],

            vertical=True,

            pills=True,

        ),

    ],

    style=SIDEBAR\_STYLE,

)

content = html.Div(id="page-content", style=CONTENT\_STYLE)

app.layout = html.Div([dcc.Location(id="url"), sidebar, content])

@app.callback(

    Output("page-content", "children"),

    [Input("url", "pathname")])

def render\_page\_content(pathname):

    if pathname == "/":

        return all\_map.layout

    elif pathname == "/page-1":

        return country.layout

    elif pathname == "/page-2":

        return indicators.layout

    return html.Div(

        [

            html.H1("404: Not found", className="text-danger"),

            html.Hr(),

            html.P(f"The pathname {pathname} was not recognised..."),

        ],

        className="p-3 bg-light rounded-3",

    )

if \_\_name\_\_ == '\_\_main\_\_':

        app.run\_server(debug=True)

data.py

import pandas as pd

df = pd.read\_csv('https://raw.githubusercontent.com/ThuwarakeshM/geting-started-with-plottly-dash/main/life\_expectancy.csv',

                sep=',')

all\_cont = df['continent'].unique()

all\_map.py

from dash import html, dcc, callback, Output, Input

import dash\_bootstrap\_components as dbc

import plotly.express as px

from data import df

layout = dbc.Container([

    dbc.Row ([

        dbc.Col(

                html.Div([

                html.H1("Тепловая карта показателей"),

                html.P("Анализ основных показателей по странам мира с 2000 по 2015 годы."),

                html.P(" Используйте фильтр, чтобы увидеть результат."),

                html.Hr(style={'color': 'black'}),

            ], style={'textAlign': 'center'})

        )

    ]),

    html.Br(),

    dbc.Row ([

        dbc.Col([

            dbc.Label("Выберите показатель:"),

            dbc.RadioItems(

                options=[

                    {'label':'Продолжительность жизни', 'value': 'Life expectancy'},

                    {'label':'Население', 'value': 'Population'},

                    {'label':'ВВП', 'value': 'GDP'},

                    {'label':'Школьное образование', 'value': 'Schooling'},

                ],

                value='GDP',

                id='crossfilter-ind',

            ),

        ],width=3),

        dbc.Col([

            dcc.Graph(id = 'choropleth', config={'displayModeBar': False}),

        ],width=9)

    ])

])

@callback(

    Output('choropleth', 'figure'),

    Input('crossfilter-ind', 'value')

)

def update\_choropleth(indication):

    figure = px.choropleth(

        df,

        locations='Country',

        locationmode = 'country names',

        color=indication,

        hover\_name='Country',

        hover\_data = {'Country':True,'Year':True,'Status':False,

                    'Life expectancy':True,'Population':True,

                    'GDP':True,'Schooling':True,

                    'continent':False},

        labels={'Country':'Страна', 'Year':'Год',

                'Population':'Население', 'Life expectancy':'Продолжительность жизни',

                'GDP':'ВВП', 'Schooling':'Продолжительность обучения'},

        color\_continuous\_scale=px.colors.sequential.BuPu,

        animation\_frame='Year',

        )

    figure.update\_layout(margin={"r":0,"t":0,"l":0,"b":0},

                        showlegend=False)

    return figure

country.py

from dash import html, dcc, callback, Output, Input

import dash\_bootstrap\_components as dbc

import plotly.express as px

import plotly.graph\_objects as go

from data import df, all\_cont

layout = dbc.Container([

    dbc.Row ([

        dbc.Col(

                html.Div([

                html.H3("Подробная информация о выбранной стране"),

                html.Hr(style={'color': 'black'}),

            ], style={'textAlign': 'center'})

        )

    ]),

    html.Br(),

    dbc.Row ([

        dbc.Col([

            html.P("Выберите континент:")

        ],width=2),

        dbc.Col([

            dcc.Dropdown(

                id = 'crossfilter-cont',

                # заполняем дропдаун уникальными значениями континентоы из датасета

                options = [{'label': i, 'value': i} for i in all\_cont],

                # значение континента, выбранное по умолчанию

                value = all\_cont[0],

                # возможность множественного выбора

                multi = False

            )

        ],width=3),

        dbc.Col([

            html.P("Выберите страну:")

        ],width=2),

        dbc.Col([

            dcc.Dropdown(

                id = 'crossfilter-count',

                multi = False

            )

        ],width=3)

    ]),

    html.Br(),

        dbc.Row ([

            dbc.Col([

                dbc.Card([

                    dbc.Row([

                        dbc.CardHeader("Уровень жизни")

                    ]),

                    dbc.Row([

                        dbc.Col([

                            dbc.CardImg(src='/static/images/money.png')], width= 4),

                        dbc.Col([

                            dbc.CardBody(

                                html.P(

                                id='card\_text1',

                                className="card-value"),

                            )], width= 8),

                    ])

                ], color = "success", outline=True, style={'textAlign': 'center'}),

            ],width=3),

            dbc.Col([

                dbc.Card([

                    dbc.Row([

                        dbc.CardHeader("Численность населения")

                    ]),

                    dbc.Row([

                        dbc.Col([

                            dbc.CardImg(src='/static/images/religious.png')], width= 4),

                        dbc.Col([

                            dbc.CardBody(

                                html.P(

                                id='card\_text2',

                                className="card-value"),

                            )], width= 8),

                    ])

                ], color = "danger", outline=True, style={'textAlign': 'center'}),

            ],width=3),

            dbc.Col([

                dbc.Card([

                    dbc.Row([

                        dbc.CardHeader("Образование")

                    ]),

                    dbc.Row([

                        dbc.Col([

                            dbc.CardImg(src='/static/images/diploma.png')], width= 4),

                        dbc.Col([

                            dbc.CardBody(

                                html.P(

                                id='card\_text3',

                                className="card-value"),

                            )], width= 8),

                    ])

                ], color = "primary", outline=True, style={'textAlign': 'center'}),

            ],width=3),

            dbc.Col([

                dbc.Card([

                    dbc.Row([

                        dbc.CardHeader("Статус")

                    ]),

                    dbc.Row([

                        dbc.Col([

                            dbc.CardImg(src='/static/images/globe.png')], width= 4),

                        dbc.Col([

                            dbc.CardBody(

                                html.P(

                                id='card\_text4',

                                className="card-value"),

                            )], width= 8),

                    ])

                ], color = "info", outline=True, style={'textAlign': 'center'}),

],width=3)

    ]),

    html.Br(),

    dbc.Container([

        dbc.Row ([

            dbc.Col([

                html.H5("Уровень ВВП по странам континента 2015 год"),

                dcc.Graph(id='choropleth1', config={'displayModeBar':False}),

            ],width=8),

            dbc.Col([

                dbc.Row([

                    html.Div(id="card1"),

                ]),

                html.Br(),

                dbc.Row([

                    html.H5("ТОП-5 стран по ВВП"),

                    html.Div(id="table1"),

                ], style={'textAlign':'center'})

            ],width=4)

        ], style={'textAlign':'center'}),

    ])

])

@callback(

    [Output('crossfilter-count', 'options'),

    Output('crossfilter-count', 'value'),

    ],

    Input('crossfilter-cont', 'value')

)

def update\_region(cont):

    all\_count=df[(df['continent'] == cont)]['Country'].unique()

    dd\_count = [{'label': i, 'value': i} for i in all\_count]

    dd\_count\_value = all\_count[0]

    return dd\_count, dd\_count\_value

@callback(

    [Output('card\_text1','children'),

    Output('card\_text2','children'),

    Output('card\_text3','children'),

    Output('card\_text4','children'),

    Output('card1','children'),

    Output('table1', 'children'),

    Output('choropleth1', 'figure')

    ],

    [Input('crossfilter-count', 'value'),

    Input('crossfilter-cont', 'value'),

    ]

)

def update\_card(count, cont):

    df\_count=df[(df['Country'] == count)&(df['Year'] == 2015)]

    df\_count14=df[(df['Country'] == count)&(df['Year'] == 2014)]

    gdp\_count=df[(df['continent'] == cont)&(df['Year'] == 2015)].sort\_values(by='GDP', ascending=False)

    ct1=df\_count.iloc[0]['Life expectancy']

    ct2=df\_count.iloc[0]['Population']

    ct3=df\_count.iloc[0]['Schooling']

    ct4=df\_count.iloc[0]['Status']

    gdp15=df\_count.iloc[0]['GDP']

    gdp14=df\_count14.iloc[0]['GDP']

    gdp\_table=gdp\_count.iloc[0:5][['Country','GDP']]

    delta\_gdp=round((gdp15-gdp14)/gdp14, 2)\*100

    card1 = dbc.Card([

        dbc.Row([

            dbc.CardHeader("Валовый внутренний продукт"),

        ]),

        dbc.Row([

            html.Div([

                html.H5(gdp15),

                html.Div([

                    html.I(delta\_gdp),

                ]),

                html.I(' в % к 2014 году'),

            ])

        ],)

    ], style={'textAlign': 'center'})

    table = dbc.Table.from\_dataframe(

        gdp\_table, striped=True, bordered=True, hover=True, index=False)

    figure = px.choropleth(

        gdp\_count,

        locations='Country',

        locationmode = 'country names',

        color="GDP",

        hover\_name='Country',

        hover\_data = {'Country':True,'Year':False,'Status':False,

                    'Life expectancy':True,'Population':True,

                    'GDP':True,'Schooling':False,

                    'continent':False},

        labels={'Country':'Страна', 'Year':'Год',

                'Population':'Население', 'Life expectancy':'Продолжительность жизни',

                'GDP':'ВВП', 'Schooling':'Продолжительность обучения'},

        color\_continuous\_scale=px.colors.sequential.RdPu,

        )

    figure.update\_layout(margin={"r":0,"t":0,"l":0,"b":0},

                        showlegend=False,

                        coloraxis\_showscale=False)

    return f"{ct1} лет", f"{ct2} чел.", f"{ct3} лет", ct4, card1, table, figure

Реализовано:

1. Смена цветовой палитры
2. Добавлены единицы измерения для 4 карточек на странице «страны»
3. Изменена тема на “Cyborg”

Не реализовано:

1. Страница «Статистика»