|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Кафедра патологической физиологии имени В.В. Пашутина** | | | | | | | | |
| **ЛАБОРАТОРИЯ ИННОВАЦИОННЫХ ОБРАЗОВАТЕЛЬНЫХ  ТЕХНОЛОГИЙ** | | | | | | | | |
|  | | | | | | | | |
|  | | | | | | |  | |
|  | | | | | | | | |
|  | | | | | | | | |
| ИСХОДНЫЙ КОД | | | | | | | | |
| программы RatRopy | | | | | | | | |
|  | | | | | | | | |
| **ЛИОТ-ПО-01.ИК** | | | | | | | | |
|  | | | | | | | | |
|  | | | | | | | | |
|  | | | | | |  | |  |
|  | | | | | |  | |  |
|  | | | | | | | | |
|  | Изм. | № док. | Подп. | Дата |  | | | |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  | | | | | | | | |
| **2025** | | | | | | | | |

import flet as ft

import math

import datetime

from collections import defaultdict

import sqlite3

from fpdf import FPDF

import os

import json

import asyncio

# Инициализация базы данных

def init\_db():

conn = sqlite3.connect('ratropy.db')

cursor = conn.cursor()

cursor.execute('''

CREATE TABLE IF NOT EXISTS animals (

id TEXT PRIMARY KEY,

species TEXT,

age TEXT,

weight TEXT,

info TEXT

)''')

cursor.execute('''

CREATE TABLE IF NOT EXISTS experiments (

id INTEGER PRIMARY KEY AUTOINCREMENT,

animal\_id TEXT,

date TEXT,

attempt TEXT,

events TEXT,

entropy REAL,

FOREIGN KEY (animal\_id) REFERENCES animals (id)

)''')

conn.commit()

conn.close()

init\_db()

# Функции работы с БД

def add\_animal\_db(animal):

conn = sqlite3.connect('ratropy.db')

cursor = conn.cursor()

cursor.execute('''

INSERT INTO animals (id, species, age, weight, info)

VALUES (?, ?, ?, ?, ?)

''', (animal['id'], animal['species'], animal['age'], animal['weight'], animal['info']))

conn.commit()

conn.close()

def update\_animal\_db(animal):

conn = sqlite3.connect('ratropy.db')

cursor = conn.cursor()

cursor.execute('''

UPDATE animals

SET species = ?, age = ?, weight = ?, info = ?

WHERE id = ?

''', (animal['species'], animal['age'], animal['weight'], animal['info'], animal['id']))

conn.commit()

conn.close()

def get\_animals\_db():

conn = sqlite3.connect('ratropy.db')

cursor = conn.cursor()

cursor.execute('SELECT \* FROM animals')

animals = [dict(zip(['id', 'species', 'age', 'weight', 'info'], row)) for row in cursor.fetchall()]

conn.close()

return animals

def get\_animal\_db(animal\_id):

conn = sqlite3.connect('ratropy.db')

cursor = conn.cursor()

cursor.execute('SELECT \* FROM animals WHERE id = ?', (animal\_id,))

row = cursor.fetchone()

conn.close()

if row:

return dict(zip(['id', 'species', 'age', 'weight', 'info'], row))

return None

def add\_experiment\_db(experiment):

conn = sqlite3.connect('ratropy.db')

cursor = conn.cursor()

cursor.execute('''

INSERT INTO experiments (animal\_id, date, attempt, events, entropy)

VALUES (?, ?, ?, ?, ?)

''', (

experiment['animal\_id'],

experiment['date'],

experiment['attempt'],

json.dumps(experiment['events']),

experiment['entropy']

))

conn.commit()

conn.close()

def get\_experiments\_db():

conn = sqlite3.connect('ratropy.db')

cursor = conn.cursor()

cursor.execute('''

SELECT experiments.\*, animals.species

FROM experiments

LEFT JOIN animals ON experiments.animal\_id = animals.id

''')

experiments = []

for row in cursor.fetchall():

exp = dict(zip(

['id', 'animal\_id', 'date', 'attempt', 'events', 'entropy', 'species'],

row

))

exp['events'] = json.loads(exp['events'])

experiments.append(exp)

conn.close()

return experiments

def get\_experiment\_db(experiment\_id):

conn = sqlite3.connect('ratropy.db')

cursor = conn.cursor()

cursor.execute('''

SELECT experiments.\*, animals.species

FROM experiments

LEFT JOIN animals ON experiments.animal\_id = animals.id

WHERE experiments.id = ?

''', (experiment\_id,))

row = cursor.fetchone()

conn.close()

if row:

exp = dict(zip(

['id', 'animal\_id', 'date', 'attempt', 'events', 'entropy', 'species'],

row

))

exp['events'] = json.loads(exp['events'])

return exp

return None

# Основное приложение

def main(page: ft.Page):

page.title = "RatRopy Explorer"

page.theme\_mode = ft.ThemeMode.DARK # Исправлено на темную тему

page.padding = 20

page.scroll = ft.ScrollMode.AUTO

# Глобальные переменные состояния

current\_experiment = None

timer\_running = False

start\_time = None

timer\_text = ft.Text("00:00:00", size=24)

events\_list = ft.Ref[ft.ListView]()

selected\_animal\_id = None

BEHAVIORS = [

"Горизонтальное положение",

"Центральная стойка",

"Переферическая стойка",

"Груминг",

"Замирание",

"Заглядывание в норки",

"Обнюхивание"

]

# Расчет энтропии

def calculate\_entropy(events):

acts = [e[1] for e in events]

n = len(acts)

if n == 0:

return 0

# Вероятности отдельных актов

p\_i = defaultdict(int)

for act in acts:

p\_i[act] += 1

for k in p\_i:

p\_i[k] /= n

# Вероятности пар

p\_ij = defaultdict(lambda: defaultdict(int))

counts\_j = defaultdict(int)

for i in range(1, n):

prev = acts[i-1]

curr = acts[i]

p\_ij[prev][curr] += 1

counts\_j[prev] += 1

# Вероятности троек

p\_ijk = defaultdict(lambda: defaultdict(lambda: defaultdict(int)))

counts\_ij = defaultdict(lambda: defaultdict(int))

for i in range(2, n):

prev2 = acts[i-2]

prev1 = acts[i-1]

curr = acts[i]

p\_ijk[prev2][prev1][curr] += 1

counts\_ij[prev2][prev1] += 1

# Вычисление энтропии

H1 = 0

for pi in p\_i.values():

if pi > 0:

H1 -= pi \* math.log2(pi)

H2 = 0

for j, transitions in p\_ij.items():

for count in transitions.values():

p = count / counts\_j[j]

if p > 0:

H2 -= p \* math.log2(p)

H3 = 0

for i, row in p\_ijk.items():

for j, transitions in row.items():

for count in transitions.values():

p = count / counts\_ij[i][j]

if p > 0:

H3 -= p \* math.log2(p)

return H1 + H2 + H3

# Генерация PDF отчета

def generate\_pdf(experiment, animal):

pdf = FPDF()

pdf.add\_page()

pdf.set\_font("Arial", size=12)

# Заголовок

pdf.cell(0, 10, f"Отчет об исследовании - {experiment['date']}", 0, 1, 'C')

pdf.ln(10)

# Паспортные данные

pdf.cell(0, 10, "Паспортная часть:", 0, 1)

pdf.cell(0, 10, f"Животное: {animal['id']} ({animal['species']})", 0, 1)

pdf.cell(0, 10, f"Возраст: {animal['age']}, Вес: {animal['weight']}", 0, 1)

pdf.ln(5)

# Результаты

pdf.cell(0, 10, f"Показатель энтропии: {experiment['entropy']:.4f}", 0, 1)

pdf.ln(10)

# История событий

pdf.cell(0, 10, "История поведенческих актов:", 0, 1)

for time, event in experiment['events']:

pdf.cell(0, 10, f"{time} - {event}", 0, 1)

# Сохранение файла

filename = f"report\_{animal['id']}\_{experiment['date'].replace(' ', '\_')}.pdf"

pdf.output(filename)

return filename

# Диалог сохранения файла

def save\_file\_result(e: ft.FilePickerResultEvent):

if e.path:

animal = get\_animal\_db(current\_experiment['animal\_id'])

filename = generate\_pdf(current\_experiment, animal)

os.rename(filename, e.path)

page.snack\_bar = ft.SnackBar(ft.Text(f"Отчет сохранен: {e.path}"))

page.snack\_bar.open = True

page.update()

file\_picker = ft.FilePicker(on\_result=save\_file\_result)

page.overlay.append(file\_picker)

# Форматирование времени

def format\_time(seconds):

return str(datetime.timedelta(seconds=seconds))

# Обновление таймера

# Замените функцию update\_timer на следующую версию:

async def update\_timer():

nonlocal timer\_running

while timer\_running:

elapsed = datetime.datetime.now() - start\_time

seconds = int(elapsed.total\_seconds())

timer\_text.value = format\_time(seconds)

# Обновляем всю страницу, а не только текстовый элемент

await page.update\_async()

await asyncio.sleep(0.1)

# Обработчики поведения

def behavior\_clicked(behavior):

if timer\_running:

elapsed = datetime.datetime.now() - start\_time

seconds = int(elapsed.total\_seconds())

time\_str = format\_time(seconds)

current\_experiment['events'].append((time\_str, behavior))

events\_list.current.controls.append(

ft.ListTile(

title=ft.Text(behavior),

subtitle=ft.Text(time\_str)

)

)

events\_list.current.update()

# Завершение исследования

def finish\_experiment(e):

nonlocal timer\_running

timer\_running = False

current\_experiment['entropy'] = calculate\_entropy(current\_experiment['events'])

add\_experiment\_db(current\_experiment)

show\_report(current\_experiment)

# Начало исследования

def start\_experiment(e):

nonlocal current\_experiment, timer\_running, start\_time

date = date\_input.value

animal\_id = animal\_id\_input.value

attempt = attempt\_input.value

if not all([date, animal\_id, attempt]):

page.snack\_bar = ft.SnackBar(ft.Text("Заполните все поля!"))

page.snack\_bar.open = True

page.update()

return

animal = get\_animal\_db(animal\_id)

if not animal:

page.snack\_bar = ft.SnackBar(ft.Text(f"Животное с ID {animal\_id} не найдено!"))

page.snack\_bar.open = True

page.update()

return

current\_experiment = {

'animal\_id': animal\_id,

'date': date,

'attempt': attempt,

'events': [],

'entropy': None

}

start\_time = datetime.datetime.now()

timer\_running = True

# Clear previous events

if events\_list.current:

events\_list.current.controls.clear()

page.go("/experiment")

page.update()

# Запускаем таймер через page.run\_task (передаем функцию, а не её вызов)

page.run\_task(update\_timer)

# Показать отчет

def show\_report(experiment):

animal = get\_animal\_db(experiment['animal\_id'])

report\_content.controls = [

ft.Divider(),

ft.Text(f"Дата: {experiment['date']}"),

ft.Text(f"Животное: {animal['id']} ({animal['species']})"),

ft.Text(f"Попытка: {experiment['attempt']}"),

ft.Text(f"Энтропия: {experiment['entropy']:.4f}", size=20, color="blue"),

ft.Text("История событий:", weight="bold")

]

for time, event in experiment['events']:

report\_content.controls.append(ft.Text(f"{time} - {event}"))

report\_content.controls.append(

ft.ElevatedButton(

"Экспорт в PDF",

icon=ft.Icons.PICTURE\_AS\_PDF,

on\_click=lambda \_: file\_picker.save\_file()

)

)

page.go("/report")

page.update()

# Добавление животного

def add\_animal(e):

animal = {

'id': id\_input.value,

'species': species\_input.value,

'age': age\_input.value,

'weight': weight\_input.value,

'info': info\_input.value

}

add\_animal\_db(animal)

page.snack\_bar = ft.SnackBar(ft.Text(f"Животное {animal['id']} добавлено!"))

page.snack\_bar.open = True

page.go("/")

page.update()

# Редактирование животного

def save\_animal\_changes(e):

animal = {

'id': selected\_animal\_id,

'species': edit\_species\_input.value,

'age': edit\_age\_input.value,

'weight': edit\_weight\_input.value,

'info': edit\_info\_input.value

}

update\_animal\_db(animal)

page.snack\_bar = ft.SnackBar(ft.Text("Данные сохранены!"))

page.snack\_bar.open = True

page.go("/animals")

page.update()

# UI Компоненты

# Главный экран

welcome\_view = ft.Column(

controls=[

ft.Text("RatRopy Explorer", size=30, weight="bold"),

ft.ElevatedButton("Новое исследование", on\_click=lambda \_: page.go("/new\_experiment")),

ft.ElevatedButton("Добавить животное", on\_click=lambda \_: page.go("/add\_animal")),

ft.ElevatedButton("Профиль животного", on\_click=lambda \_: page.go("/animals")),

ft.ElevatedButton("История исследований", on\_click=lambda \_: page.go("/history"))

],

spacing=20,

horizontal\_alignment=ft.CrossAxisAlignment.CENTER

)

# Форма добавления животного

id\_input = ft.TextField(label="Учетный номер")

species\_input = ft.TextField(label="Вид животного")

age\_input = ft.TextField(label="Возраст")

weight\_input = ft.TextField(label="Вес")

info\_input = ft.TextField(label="Дополнительная информация", multiline=True)

add\_animal\_view = ft.Column(

controls=[

ft.Text("Добавить животное", size=24),

id\_input,

species\_input,

age\_input,

weight\_input,

info\_input,

ft.Row([

ft.ElevatedButton("Назад", on\_click=lambda \_: page.go("/")),

ft.ElevatedButton("Сохранить", on\_click=add\_animal)

], spacing=10)

],

spacing=15

)

# Список животных

animal\_cards = ft.GridView(

expand=True,

runs\_count=3,

max\_extent=300,

child\_aspect\_ratio=3.0,

spacing=10,

run\_spacing=10

)

def update\_animal\_cards():

animal\_cards.controls.clear()

for animal in get\_animals\_db():

animal\_cards.controls.append(

ft.Card(

content=ft.Container(

content=ft.Column([

ft.ListTile(

title=ft.Text(f"ID: {animal['id']}"),

subtitle=ft.Text(animal['species']),

on\_click=lambda e, a=animal['id']: show\_animal\_detail(a)

)

])

)

)

)

animals\_view = ft.Column(

controls=[

ft.Text("Профиль животного", size=24),

animal\_cards,

ft.ElevatedButton("На главную", on\_click=lambda \_: page.go("/")) # Добавлена кнопка

]

)

# Детали животного

edit\_species\_input = ft.TextField(label="Вид животного")

edit\_age\_input = ft.TextField(label="Возраст")

edit\_weight\_input = ft.TextField(label="Вес")

edit\_info\_input = ft.TextField(label="Дополнительная информация", multiline=True)

animal\_detail\_view = ft.Column()

def show\_animal\_detail(animal\_id):

nonlocal selected\_animal\_id

selected\_animal\_id = animal\_id

animal = get\_animal\_db(animal\_id)

if animal:

edit\_species\_input.value = animal['species']

edit\_age\_input.value = animal['age']

edit\_weight\_input.value = animal['weight']

edit\_info\_input.value = animal['info']

animal\_detail\_view.controls = [

ft.Row([

ft.IconButton(icon=ft.Icons.ARROW\_BACK, on\_click=lambda \_: page.go("/animals")),

ft.Text(f"Животное: {animal['id']}", size=20),

]),

ft.Text(f"Вид: {animal['species']}"),

ft.Text(f"Возраст: {animal['age']}"),

ft.Text(f"Вес: {animal['weight']}"),

ft.Text(f"Доп. информация: {animal['info']}"),

ft.ElevatedButton("Редактировать", on\_click=lambda \_: toggle\_edit\_mode(True))

]

page.go("/animal\_detail")

page.update()

def toggle\_edit\_mode(edit):

if edit:

animal\_detail\_view.controls = [

ft.Row([

ft.IconButton(icon=ft.Icons.ARROW\_BACK, on\_click=lambda \_: page.go("/animals")),

ft.Text("Редактирование животного", size=20),

]),

edit\_species\_input,

edit\_age\_input,

edit\_weight\_input,

edit\_info\_input,

ft.ElevatedButton("Сохранить", on\_click=save\_animal\_changes)

]

else:

show\_animal\_detail(selected\_animal\_id)

page.update()

# Форма нового исследования

date\_input = ft.TextField(

label="Дата",

value=datetime.datetime.now().strftime("%Y-%m-%d %H:%M")

)

animal\_id\_input = ft.TextField(label="Номер животного")

attempt\_input = ft.TextField(label="Номер попытки")

new\_experiment\_view = ft.Column(

controls=[

ft.Row([

ft.IconButton(icon=ft.Icons.ARROW\_BACK, on\_click=lambda \_: page.go("/")),

ft.Text("Новое исследование", size=24),

]),

date\_input,

animal\_id\_input,

attempt\_input,

ft.ElevatedButton("Начать исследование", on\_click=start\_experiment)

],

spacing=15

)

# Экран исследования

behavior\_buttons = []

for behavior in BEHAVIORS:

# Используем замыкание для правильной привязки поведения

def make\_behavior\_handler(b):

return lambda e: behavior\_clicked(b)

behavior\_buttons.append(

ft.ElevatedButton(

behavior,

on\_click=make\_behavior\_handler(behavior),

width=200,

height=60

)

)

experiment\_view = ft.Column(

controls=[

ft.Row([

ft.IconButton(icon=ft.Icons.ARROW\_BACK, on\_click=lambda \_: page.go("/")),

ft.Text("Исследование", size=24),

]),

ft.Row([

ft.Column([

ft.Text("Паспортная часть", weight="bold"),

ft.Text(f"Животное: {current\_experiment['animal\_id'] if current\_experiment else ''}"),

ft.Text(f"Попытка: {current\_experiment['attempt'] if current\_experiment else ''}")

]) if current\_experiment else ft.Text(""),

ft.ElevatedButton("Завершить", color="red", on\_click=finish\_experiment)

], alignment=ft.MainAxisAlignment.SPACE\_BETWEEN),

#ft.Container(

#content=timer\_text,

#alignment=ft.alignment.center,

#padding=20

#),

ft.Text("Поведенческие акты:", weight="bold"),

ft.Row(

controls=behavior\_buttons,

wrap=True,

spacing=10,

run\_spacing=10,

scroll=ft.ScrollMode.AUTO

),

ft.Text("История событий:", weight="bold"),

ft.Container(

content=ft.ListView(ref=events\_list, expand=True),

border=ft.border.all(1, ft.Colors.GREY\_700),

border\_radius=10,

padding=10,

expand=True

)

],

scroll=ft.ScrollMode.AUTO,

expand=True

)

# Отчет об исследовании

report\_content = ft.Column()

report\_view = ft.Column(

controls=[

ft.Text("Отчет об исследовании", size=24),

report\_content,

ft.ElevatedButton("На главную", on\_click=lambda \_: page.go("/"))

]

)

# История исследований

history\_cards = ft.ListView(expand=True)

def update\_history\_cards():

history\_cards.controls.clear()

for exp in get\_experiments\_db():

history\_cards.controls.append(

ft.Card(

content=ft.Container(

content=ft.ListTile(

title=ft.Text(f"Дата: {exp['date']}"),

subtitle=ft.Text(f"Животное: {exp['animal\_id']} ({exp.get('species', 'N/A')})"),

trailing=ft.Text(f"Энтропия: {exp['entropy']:.4f}"),

on\_click=lambda e, ex=exp['id']: show\_experiment\_report(ex)

),

padding=10

)

)

)

def show\_experiment\_report(exp\_id):

experiment = get\_experiment\_db(exp\_id)

if experiment:

show\_report(experiment)

history\_view = ft.Column(

controls=[

ft.Text("История исследований", size=24),

history\_cards,

ft.ElevatedButton("На главную", on\_click=lambda \_: page.go("/"))

]

)

# Маршрутизация

def route\_change(route):

page.views.clear()

page.views.append(ft.View("/", [welcome\_view]))

if page.route == "/add\_animal":

page.views.append(ft.View("/add\_animal", [add\_animal\_view]))

elif page.route == "/animals":

update\_animal\_cards()

page.views.append(ft.View("/animals", [animals\_view]))

elif page.route == "/animal\_detail":

page.views.append(ft.View("/animal\_detail", [animal\_detail\_view]))

elif page.route == "/new\_experiment":

page.views.append(ft.View("/new\_experiment", [new\_experiment\_view]))

elif page.route == "/experiment":

page.views.append(ft.View("/experiment", [experiment\_view]))

elif page.route == "/report":

page.views.append(ft.View("/report", [report\_view]))

elif page.route == "/history":

update\_history\_cards()

page.views.append(ft.View("/history", [history\_view]))

page.update()

page.on\_route\_change = route\_change

page.go(page.route)

if \_\_name\_\_ == "\_\_main\_\_":

ft.app(target=main)