ҚАЗАҚСТАН РЕСПУБЛИКАСЫНЫҢ БІЛІМ ЖӘНЕ ҒЫЛЫМ МИНИСТРЛІГІ

СӘТБАЕВ УНИВЕРСИТЕТІ

Автоматика және Информациялық Технологиялар институты

Программалық Инженерия кафедрасы



ЛАБОРАТОРИЯЛЫҚ ЖҰМЫС #6

Тақырыбы: Функционалды реактивті бағдарламалау (FRP)

|  |  |  |  |
| --- | --- | --- | --- |
| № | Жұмысты орындау сапасы | Баға диапазоны | Орындалған  % |
| 1 | Орындалған жоқ | 0% |  |
| 2 | Орындалды | 0-50% |  |
| 3 | Материялдық өзіндік жүйелендіру | 0-10% |  |
| 4 | Талап етілген көлемде және көрсетілген мерзімде орындау | 0-5% |  |
| 5 | Қосымша ғылыми әдебиеттерді пайдалану | 0-5% |  |
| 6 | Орындаған тапсырманың ерекшелігі | 0-10% |  |
| 7 | СӨЖ-ді қорғау | 0-20% |  |
|  | Қорытынды: | 0-100% |  |

Оқытушы: Шаяхметов Д

Студент: Ұлдақан А

Мамандығы: Computer Science

Тобы: Дс 7:50 – 9:45

Алматы 2024 ж

**Мақсат:**

Python контекстінде функционалдық реактивті бағдарламалау (FRP) принциптерін үйреніңіз және қолданыңыз. Жұмыстың мақсаты - деректер ағындарын, реактивті айнымалыларды және асинхронды оқиғаларды өңдеуді қоса алғанда, FRP негізгі тұжырымдамаларын түсіну және жауап беретін және модульдік қосымшаларды жасау үшін осы тұжырымдамаларды пайдалану дағдыларын дамыту.

Тапсырмалар:

1. FRP негіздері:

- Ағындар, сигналдар және реактивті айнымалылар сияқты негізгі FRP тұжырымдамаларын үйреніңіз.

- реактивті өзгерістерді тарату және тәуелділікті басқару механизмдерін талдау.

2. Қарапайым FRP сценарийлерін іске асыру:

- өзгермелі деректерді өңдеу және көрсетумен байланысты мәселелерді шешу үшін FRP қолдану.

- Пайдаланушы әрекеттеріне немесе сыртқы оқиғаларға жауап ретінде реактивті деректерді жаңартуды көрсететін мысалдарды әзірлеу.

3. Қолданыстағы қолданбалармен интеграция:

- Қолданыстағы Python қолданбаларына FRP тәсілдерін біріктіру жолдарын қарастыру.

- FRP қолдану арқылы жауап беруді жақсарту және код күрделілігін азайту мүмкіндіктерін талдау.

4. FRP-дегі қосымша тақырыптар:

- Қателерді өңдеу, ағынды біріктіру және асинхронды операцияларды қоса алғанда, FRP жүйесіндегі озық әдістер мен үлгілерді үйреніңіз.

- көптеген деректер көздерін тиімді өңдеуге қабілетті күрделі реактивті жүйелерді жасау.

5. Сыни талдау және рефлексия:

- Нақты әлемдегі қолданбалар контекстінде FRP артықшылықтары мен шектеулерін бағалау.

- Нақты бағдарламалық тапсырмалар үшін тәсілдер мен құралдарды таңдауда сыни тұрғыдан ойлауды дамыту.

Зертханалық жұмыстың маңыздылығы:

Зертхана студенттерді қазіргі заманғы бағдарламалық қамтамасыз етуді әзірлеуде маңыздырақ болып келе жатқан функционалдық реактивті бағдарламалау парадигмасымен таныстыруға арналған. FRP интерактивті және асинхронды қолданбаларды құруға арналған талғампаз шешімдерді ұсынады, бұл күй мен деректер ағынын басқаруды жеңілдетеді. Жұмыс реактивті жүйелерді терең түсінуді дамытады және анағұрлым жауап беретін және модульдік қосымшаларды жобалау дағдыларын жетілдіреді.

**Жеке тапсырмалар:**

Әрбір студентке топ тізіміндегі санына сәйкес бірегей тапсырма беріледі (SSO қараңыз).



Функционалды реактивті бағдарламалау (FRP) өзгерістерге жауап беретін жүйелерді құру үшін реактивті және функционалды бағдарламалау идеяларын біріктіреді.

2. FRP істер тізімі

- Тапсырмаларды қосуға және жоюға жауап беретін FRP көмегімен қарапайым істер тізімі қолданбасын жасаңыз.

**Code:**

import tkinter as tk

from rx import subject

class TodoList:

    def \_\_init\_\_(self):

*self*.tasks = []

*self*.subject = subject.Subject()

    def add\_task(self, task):

*self*.tasks.append(task)

*self*.subject.on\_next(*self*.tasks)

    def remove\_task(self, task\_index):

        del *self*.tasks[task\_index]

*self*.subject.on\_next(*self*.tasks)

    def get\_tasks(self):

        return *self*.subject

class TodoApp:

    def \_\_init\_\_(self, todo\_list, root):

*self*.todo\_list = todo\_list

*self*.root = root

*self*.setup\_ui()

    def setup\_ui(self):

*self*.task\_entry = tk.Entry(*self*.root)

*self*.task\_entry.pack()

*self*.add\_button = tk.Button(*self*.root, text="Add Task", command=*self*.add\_task)

*self*.add\_button.pack()

*self*.tasks\_label = tk.Label(*self*.root, text="Tasks:")

*self*.tasks\_label.pack()

*self*.tasks\_text = tk.Text(*self*.root, height=10, width=30)

*self*.tasks\_text.pack()

*self*.task\_click\_subject = subject.Subject()

*self*.tasks\_text.bind("<Button-1>", *self*.on\_task\_click)

*self*.todo\_list.get\_tasks().subscribe(*self*.update\_ui)

    def update\_ui(self, tasks):

*self*.tasks\_text.delete('1.0', tk.END)

        for i, task in enumerate(tasks):

*self*.tasks\_text.insert(tk.END, f"{i + 1}. {task}\n")

    def add\_task(self):

        task = *self*.task\_entry.get()

        if task:

*self*.todo\_list.add\_task(task)

*self*.task\_entry.delete(0, tk.END)

    def on\_task\_click(self, event):

        index = *self*.tasks\_text.index(tk.CURRENT)

        task\_index = int(index.split('.')[0]) - 1

*self*.task\_click\_subject.on\_next(task\_index)

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

    todo\_list = TodoList()

    root = tk.Tk()

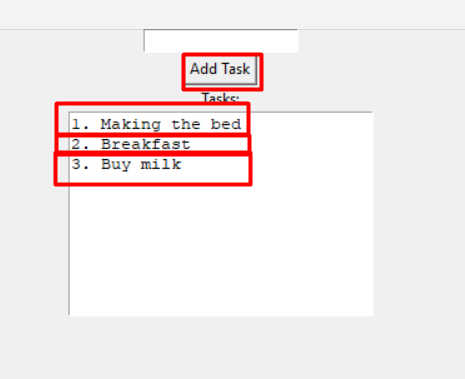
    todo\_app = TodoApp(todo\_list, root)

    todo\_app.task\_click\_subject.subscribe(lambda task\_index: todo\_list.remove\_task(task\_index))

    root.mainloop()

**Output:**

**Add tasks:**



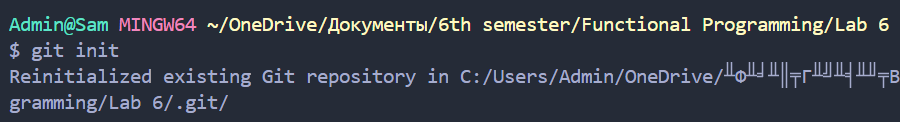
**Remove tasks:**

Изображение выглядит как текст, снимок экрана, Шрифт, дизайн

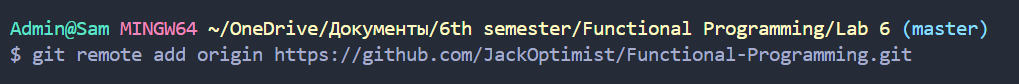
Автоматически созданное описание

**Work with GIT**

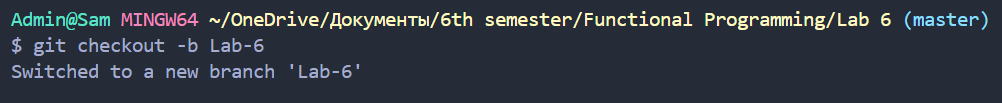
1. Initialize git repository



1. Add remote my course repository (for my example GitHUB)



1. Create new branch and switched to a new branch



1. Our python file is untracked

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

1. Now our file is tracked

Изображение выглядит как текст, Шрифт, снимок экрана

Автоматически созданное описание

1. Commit

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

1. I pushed files to remote repository (correct branch Lab-6)

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

1. Remote repository – GitHUB (My GitHUB Profile)

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание