Text

Description automatically generated

Управление на

Софтуерни Проекти

Проект

Система за организиране на задачи

Факултет по Изчислитена Техника и

Автоматизация

Катедра:

Софтуерни и Интернет Технологии

Изготвили:

Николай Николаев Панев: Факултетен номер: 20621511

Венелин Живков Дачев: Факултетен номер: 20621502

Димитър Пламенов Тодоров: Факултетен номер: 20621501

Проверил: Специалност: СИТ

ас. Даниела Петрова Курс: 3 .............................................. Група: 1а

Задание:

Система за организиране на задачите на потребител. Системата трябва да предоставя следните възможности:

* въвеждане на нова задача – напр. описание на задачата, краен срок и др.;
* известие за началото на дадена задача;
* известие за наближаващ краен срок на задача;

Организационна Документация

Система за организация на работата (Trello):

Създадени board-ове:

* Back-End

Задачи за вършене по back-end модулите на проекта

* + To Do
  + Doing
  + Done
* Front-End

Задачи за вършене по front-end модулите на проекта

* + To Do
  + Doing
  + Done
* Database

Задачи свързани с базата данни (файлова система)

* + To Do
  + Doing
  + Done
* Brainstorm

Тук се предлагат идеи, отбелязва се кои ще бъдат обсъдени и кои идеи са приети

* + Brainstorm
  + To Discuss
  + Accepted
* Documentation

Това е най-активният board, понеже всяко едно действие във всеки един board бива записано в документацията. Тук също има и доста автомация на „картите“, която да подпомага следенето на прогреса по отделните компоненти на документацията.

* + Technical Documentation
  + Database
  + Project Management
  + To Edit
  + To Merge To File
  + Merged

Система за Version Control(GIT/GITHUB):

Изборът на тази система е изключително лесен, защото е технология, често използвана от, и позната на всички участници в проекта(Николай и Венелин).

Каченият проект може да намерите на този линк:

<https://github.com/NikolayNPanev/py-nvy-task>

Избор на технология (Python & CustomTkinter):

Избрахме тази технология, защото искаме да използваме нови за нас технологии, да научим техните силни и слаби страни и да разширим знанията си. Това че изполваме непозната за нас технолофия е изчислен риск, който решихме да поемем.

Тази технология позволява създаването на Python приложение с графичен интерфейс, използвайки библиотеката CustomTkinter, която е потребителско разширение на вече въществъващата в Python библиотека Tkinter.

Роли на участниците в проекта:

* Николай: Back-End & Dev-Ops & Management
* Венелин: Front-End & QA
* Димитър: Data Specialist(File system)

Крайни Срокове:

Концепция – 1 работен ден

Организация – 5 работни дни

Проектиране – 4 работни дни

Дизайн – 2 работни дни

Прототипиране – 5 работни дни

Документация – 3 работни дни

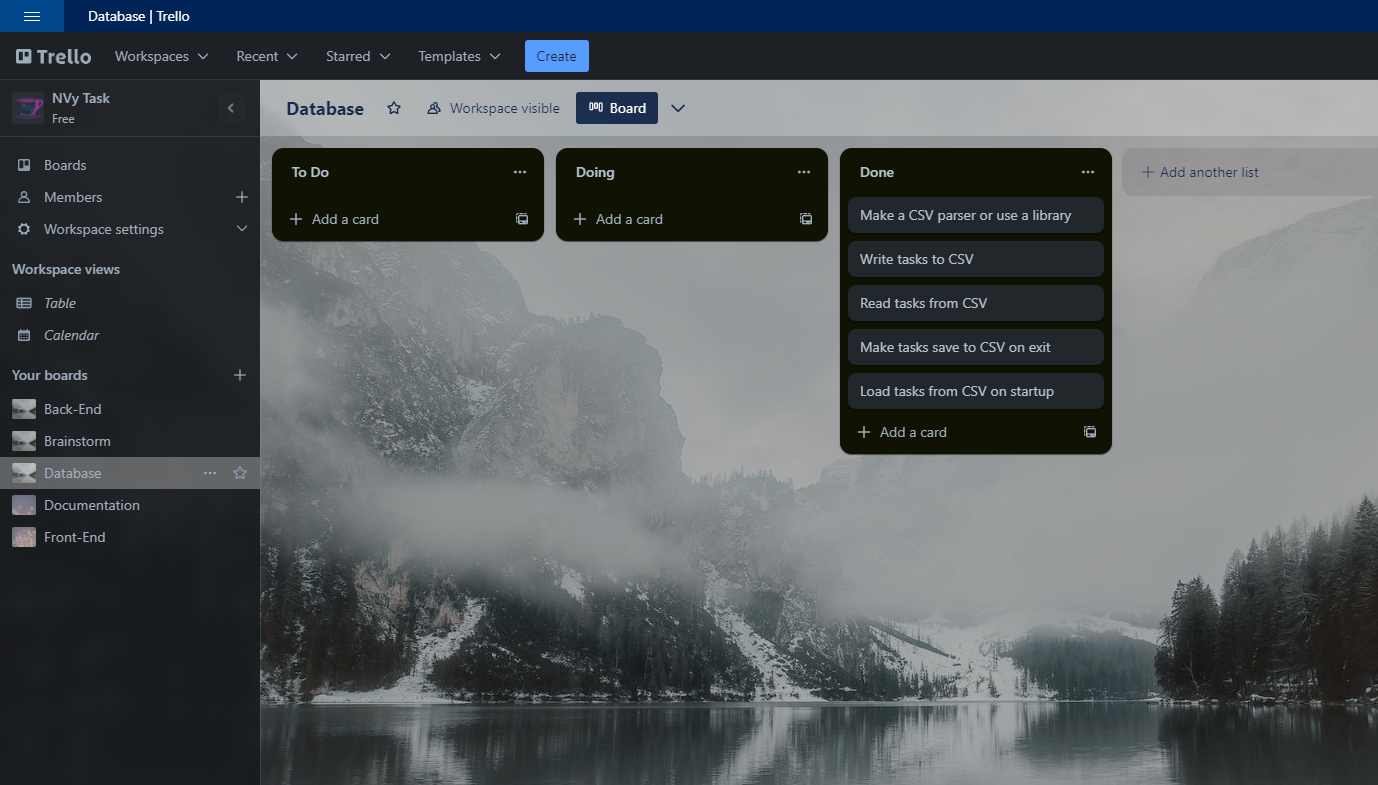
MS Project:

A screenshot of a computer

Description automatically generated with low confidence

Trello:A screenshot of a computer

Description automatically generated



A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

GitHub:

Използва се за Version Control

* Създадено Репо
* Качване на прогрес

Методи за разработка

Waterfall

Използваме метода на водопада, защото е подходящ за целта и времето за разработка на проекта. Подходът се състои от линейна последователност от фази: изисквания, разработване,тестване

Предимства на подхода:

Предвидимост, структурираност и последователност. Идеалните условия за разработването на този проект.

Недостатъци:

Ограничена способност за адаптация към промени.

Съпоставка с AGILE:

Въпреки по-голямата гъвкавост към промени на AGILЕ, подходът на водопада предоставя високо ниво на структурираност, което помага на участниците в проекта по-лесно да разпределят задачите си в ранен етап от жизненият цикъл на проекта. Също, това е перфектният подход за приложение с ясно дефинирани потребности.

Бюджет(MS Project):

A screenshot of a graph

Description automatically generated with low confidence

A picture containing text, screenshot, font, number

Description automatically generated

Трудности в изпълнението на задачата:

Основните затруднения в изпълнението на този проект са външните фактори, които засегнаха всички участници от екипа по един или друг начин. От техническа гледна точка, доста сериозно затруднение остана и използването на нова технология за създаването на графичният интерфейс на приложението, но това е риск който бе поет със съгласието на всички участници.

Техническа Документация

Тегнологии използвани за изработване на графичният интерфейс:

CustomTKinter – това е библиотека, създадена от трето лице, в Python, използвана за бързо и лесно създаване на семпли и модерни графични интерфейси. Тази библиотека пряко наследява библиотеката, TKinter, която е част от стандартният набор на библиотеки в Python.

Модел на графичният интерфейс:

Графичният интерфейс бива прост, но функционален, позволяващ лесно ориентиране на потребителя в приложението, и в леснотата си позволява на по-обхватен кръг от потребители да се възползват от неговите функционалности.

За създаване на задача, потребителят трябва да въведе 5 потребителски входа, а именно:

* Име на задача
* Крайна дата
  + Ден
  + Месец
  + Година
* Дали задачате е спешна или не
* Да натисне бутонът за създаване на задачата

След създаването на задачата, в прозорецът на приложенито

тя бива изведена с всичките си данни(Име, спешност, крайна дата). Освен данните за задачата, на същият ред се извежда и бутон за завършване на задачата. Ако потребителят реши да „завърши“ някоя вече съществуваща задача, тя бива премахната от списъчният изглед на програмата(Фиг.1).

Ако крайният срок на задача предстои в срок от 7 дни или по-малко, нейният краен срок бива изобразен в жълт цвят(Фиг.1). Крайният срок на спешните задачи също се изобразява в жълт цвят(Фиг.1), независимо след какъв период биват те. Ако крайният срок на някоя задача е пропуснат, той бива изобразен в червен цвят(Фиг.1), независимо от спешността на задачата.

A screenshot of a computer

Description automatically generated

Фиг. 1

Технологии използвани за запазване на задачи във файлова система:

За лесно запазване на създадените от потребителя задачи, приложението използва библиотека CSV, която бива част от стандартният набор от библиотеки в Python.

При всяко зареждане на приложението, от директията от която е пуснато се зарежда файл с името Tasks.csv. Ако този файл не съществува, той бива създаден при затваряне на приложението, запазвайки всички задачи създадени от потребителя по време на използването на приложението, ако има такива.

Защо CSV?

CSV(Comma Separated Values), е конвенция за записване на таблични файлове във формата на текстови, всяка колона на които бива разделена със запетая, а всеки ред със знак за нов ред. Това позволява лесна интеграция с други приложения, API, и дори създаването на помощни програми, разширяващи функционалността на NVy Task.

Класове

В приложението участват три класа за създаване на обекти:

-Task()

-App()

-AppFrame(App)

Task():

Конструктор:

def \_\_init\_\_(self,taskName, isDone:bool = False, urgency = "Normal",deadline="1/1/1970"):

        self.name = taskName

        self.isDone = isDone

        self.urgency = urgency

        self.deadline = deadline

        self.ParseDeadline()

        print(self)

Методи:

Парсиране на дати до променливи (ден, месец и година)

def ParseDeadline(self):

        dline = self.deadline.split("/")

        self.day=dline[0]

        self.month=dline[1]

        self.year=dline[2]

Метод за сравнение на задачи

def \_\_eq\_\_(self,name):

        return self.name == name

Метод за извеждане на задача като текст в конзолата(при дебъгване)

def \_\_str\_\_(self) -> str:

        str=f"Task: {self.name}"

        if(self.isDone == "True"):

            str= str+(" : Done!")

        else:

            str= str+(" : Not Done!")

        if(self.urgency == "Urgent"):

            str= str+(" : Urgent!!!")

        if(self.urgency == "Normal"):

            str= str+(" : Normal")

        return str

App(customtkinter.CTk):

Конструктор

def \_\_init\_\_(self):

        super().\_\_init\_\_()

        self.title("NVy Task")

        self.minsize(607, 800)

        self.grid\_rowconfigure(0, weight=1)

        self.grid\_columnconfigure(0, weight=1)

        self.app\_frame = AppFrame(master=self, width=300, height=200, corner\_radius=0, fg\_color="transparent")

        self.app\_frame.grid(row=0, column=0, sticky="nsew")

AppFrame(customtkinter.CTkScrollableFrame):

Конструктор

def \_\_init\_\_(self, master, \*\*kwargs):

        if(work!="False"):

            super().\_\_init\_\_(master, \*\*kwargs)

            self.grid\_rowconfigure(1, weight=0)

            self.grid\_columnconfigure((0, 1), weight=1)

            self.row=0

            self.namelable=customtkinter.CTkLabel(master=self,text="ENTER TASK")

            self.namelable.grid(row=self.row, column=0,columnspan=4)

            self.row=self.row+1

            self.textbox = customtkinter.CTkTextbox(master=self,height=20)

            self.textbox.grid(row=self.row, column=0, columnspan=4, padx=0, pady=0, sticky="ew")

            self.row=self.row+1

            self.daylable=customtkinter.CTkLabel(master=self,text="DAY")

            self.daylable.grid(row=self.row, column=0)

            self.monthlable=customtkinter.CTkLabel(master=self,text="MONTH")

            self.monthlable.grid(row=self.row, column=1)

            self.yearlable=customtkinter.CTkLabel(master=self,text="YEAR")

            self.yearlable.grid(row=self.row, column=2,columnspan=2)

            self.row=self.row+1

            self.timeday = customtkinter.CTkComboBox(master=self, values=["1","2","3","4","5","6","7","8","9","10","11","12","13","14","15","16","17","18","19","20","21","22","23","24","25","26","27","28","29","30","31"])

            self.timeday.grid(row=self.row, column=0, padx=10, pady=10, sticky="ew")

            self.timemonth = customtkinter.CTkComboBox(master=self, values=["1","2","3","4","5","6","7","8","9","10","11","12"])

            self.timemonth.grid(row=self.row, column=1, padx=10, pady=10, sticky="ew")

            years = []

            for y in range(datetime.datetime.now().year,datetime.datetime.now().year+50):

                years.append(f"{y}")

            self.timeyear = customtkinter.CTkComboBox(master=self, values=years)

            self.timeyear.grid(row=self.row, column=2,columnspan=2, padx=10, pady=10, sticky="ew")

            self.row=self.row+1

            self.urgencylable=customtkinter.CTkLabel(master=self,text="URGENCY")

            self.urgencylable.grid(row=self.row, column=0)

            self.combobox = customtkinter.CTkComboBox(master=self, values=["Normal", "Urgent"])

            self.combobox.grid(row=self.row, column=1, padx=10, pady=10, sticky="ew")

            self.button = customtkinter.CTkButton(master=self, text="Add Task", command=lambda: saveTask(f"{self.textbox.get('0.0','end').strip()}",False,f"{self.combobox.get()}",self.timeday.get()+"/"+self.timemonth.get()+"/"+self.timeyear.get()))

            self.button.grid(row=self.row, column=2, columnspan=2, padx=10, pady=10, sticky="ew")

            printAllTasks()

Функции за извеждане на екрана:

Извеждане на една задача в процореца на приложението

def showTask(task):

    global app

    taskName = customtkinter.CTkTextbox(master=app.app\_frame,height=20)

    taskName.grid(row=app.app\_frame.row+1,column=0,sticky="ew")

    taskName.insert("0.0",task.name)

    taskName.configure(state="disabled")

    if(task.urgency=="Urgent"):

        urgency = customtkinter.CTkTextbox(master=app.app\_frame,height=20,text\_color="red")

    else:

        urgency = customtkinter.CTkTextbox(master=app.app\_frame,height=20)

    urgency.grid(row=app.app\_frame.row+1,column=1,sticky="ew")

    urgency.insert("0.0",task.urgency)

    urgency.configure(state="disabled")

    late="False"

    if(int(task.year)<=int(datetime.datetime.now().year)):

        if(int(task.year)<int(datetime.datetime.now().year)):

            late="True"

        elif(int(task.month)<=int(datetime.datetime.now().month)):

            if(int(task.month)<int(datetime.datetime.now().month)):

                late="True"

            elif(int(task.day)<int(datetime.datetime.now().day)):

                late="True"

            elif(int(task.day)>int(datetime.datetime.now().day)-8):

                late="Upcoming"

    if(late=="True"):

        deadline = customtkinter.CTkTextbox(master=app.app\_frame,height=20,text\_color="red")

    elif(late=="Upcoming"):

        deadline = customtkinter.CTkTextbox(master=app.app\_frame,height=20,text\_color="yellow")

    elif(task.urgency=="Urgent"):

        deadline = customtkinter.CTkTextbox(master=app.app\_frame,height=20,text\_color="yellow")

    else:

        deadline = customtkinter.CTkTextbox(master=app.app\_frame,height=20)

    deadline.grid(row=app.app\_frame.row+1,column=2,sticky="ew")

    deadline.insert("0.0",task.deadline)

    deadline.configure(state="disabled")

    compBtn = customtkinter.CTkButton(master=app.app\_frame,text="Complete Task", command=lambda: removeTask(app.app\_frame,task.name))

    compBtn.grid(row=app.app\_frame.row+1,column=3,sticky="ew")

    app.app\_frame.row=app.app\_frame.row+1

Извеждане на всички задачи в прозореца на приложението

def printAllTasks():

    global all\_tasks

    for task in all\_tasks:

        showTask(task)

Изтриване на всички задачи от прозореца на приложението и извеждане на всички задачи отново

def refresh():

    global app

    app.app\_frame.row=4

    for task in app.app\_frame.grid\_slaves():

        if int(task.grid\_info()["row"])>4:

            task.grid\_forget()

    printAllTasks()

Функции за запазване, зареждане и изтриване на задачи:

Зареждане на всички задачи от CSV файл

def CSVToTasks():

    global all\_tasks

    with open("Tasks.csv","r") as f:

        treader = csv.reader(f)

        header = next(treader)

        for row in treader:

            all\_tasks.append(Task(row[0],row[1],row[2],row[3]))

f.close()

Запазване на всички задачи в CSV файл

def TasksToCSV():

    global all\_tasks

    tfile = open("Tasks.csv","w")

    tfile.writelines("Task,IsDone,Urgency,Deadline\n")

    for t in all\_tasks:

        tfile.writelines(f"{t.name},{t.isDone},{t.urgency},{t.deadline}\n")

    tfile.close

Запазване на задача в глобалния буфер на приложението

ef saveTask(name,isDone,urgency,deadline):

    global all\_tasks

    for task in all\_tasks:

        if(task.name==name):

            messagebox.showinfo("Error","Task already exists!")

            return

    all\_tasks.append(Task(name,isDone,urgency,deadline))

    print(f"Task {name} saved")

    showTask(all\_tasks[-1])

Премахване на задача от глобалния буфер на приложението

def removeTask(app,name):

    ct=0

    while(ct<len(all\_tasks)):

        if(all\_tasks[ct].name==name):

            del all\_tasks[ct]

            print(f"Task {name} removed")

            refresh()

        ct=ct+1