

MAKALAH
LAPORAN PROJEK
DISUSUN GUNA UNTUK MEMENUHI TUGAS AKHIR MATA
KULIAH PEMROGRAMAN BERORIENTASI OBJEK

Dosen Pengampu:
M. Bahrul Subhki, M.Kom



Disusun Oleh:

Aldestra Bagus Wardana	(2213020238)
Muhammad Ilham Hakiki	(2213020269)
Renno Bagus Adithya	(2213020226)

KELAS 2E
PROGRAM STUDI S1 TEKNIK INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS NUSANTARA PGRI KEDIRI
TAHUN 2023

KATA PENGANTAR

Puji syukur Saya panjatkan kepada Tuhan Yang Maha Esa, karena atas limpahan rahmatnya penyusun dapat menyelesaikan makalah ini dengan tepat waktu tanpa ada halangan yang berarti sesuai dengan harapan.

Ucapan terima kasih kami sampaikan kepada Bapak M. Bahrul Subhki, M.Kom selaku dosen pengampu mata kuliah Pemrograman Berorientasi Objek yang telah membantu memberikan arahan dan pemahaman dalam penyusunan makalah proyek akhir semester ini.

Kami menyadari bahwa dalam penyusunan makalah ini masih banyak kekurangan karena keterbatasan pengetahuan kami dan kami juga memohon maaf sebesar-besarnya apabila proyek akhir ini kurang sempurna. Maka dari itu penyusun sangat mengharapkan kritik dan saran untuk menyempurnakan makalah ini. Semoga apa yang kami tulis dapat bermanfaat bagi semua pihak yang membutuhkan.

Kediri, 02 Januari 2024

Kelompok 1

BAB I

PENDAHULUAN

1.1 Latar Belakang

Dalam dunia aktivitas multitasking saat ini dan tuntutan produktivitas tinggi, manajemen waktu telah menjadi keterampilan yang penting. Dengan keterbatasan waktu dan tugas sehari-hari yang semakin kompleks, diperlukan pendekatan yang lebih terstruktur untuk memastikan efisiensi dan efektivitas. Aplikasi daftar tugas menjadi solusi penting bagi individu dan organisasi untuk mengatasi tantangan ini.

Ketika teknologi memasuki berbagai bidang kehidupan, aplikasi daftar tugas memberikan peluang untuk mengatur tugas, mengelola prioritas, dan meningkatkan produktivitas. Keputusan untuk mengembangkan aplikasi To-Do List sebagai proyek pemrograman berbasis objek dianggap sebagai langkah strategis. Pendekatan ini memungkinkan pengembangan aplikasi modular yang mudah diintegrasikan dan memiliki fleksibilitas untuk menggabungkan fungsionalitas tambahan. Hal ini memungkinkan pengguna merasakan manfaat dari solusi yang tidak hanya efisien namun juga mudah disesuaikan dengan kebutuhan individu.

1.2 Perumusan Masalah

Pada hakikatnya permasalahan yang teridentifikasi adalah manajemen waktu dan pekerjaan sehari-hari yang tidak terstruktur. Keterlambatan, kelupaan, dan kurangnya pemahaman terhadap prioritas dapat menyebabkan penurunan produktivitas. Selain itu, integrasi pribadi dan pekerjaan yang buruk dapat menciptakan ketidakseimbangan yang dapat menyebabkan stres dan kelelahan.

Oleh karena itu, perumusan masalah memerlukan pemahaman yang lebih mendalam mengenai kendala-kendala tersebut dan cara mengatasinya melalui aplikasi to-do list yang dikembangkan.

1.3 Tujuan Penelitian

Tujuan utama penelitian ini adalah merancang dan mengembangkan aplikasi to-do list yang dapat memberikan solusi optimal untuk pengelolaan waktu dan tugas. Dari perspektif pemrograman berbasis objek, tujuannya mencakup membangun struktur modular, kemampuan menangani kompleksitas, dan antarmuka yang ramah pengguna. Penelitian ini juga bertujuan untuk mempelajari dampak penggunaan aplikasi to-do list terhadap peningkatan produktivitas dan kualitas hidup pengguna

1.4 Manfaat Penelitian

- 1) Peningkatan produktivitas: Aplikasi daftar tugas diharapkan dapat meningkatkan efisiensi penyelesaian tugas pengguna, mengurangi waktu yang terbuang, dan menciptakan ruang untuk fokus pada tugas yang lebih penting.
- 2) Manajemen Stres dan Kesehatan Mental: Aplikasi ini bertujuan untuk mengurangi stres dan kecemasan yang terkait dengan beban kerja berlebihan dengan membantu pengguna mengatur dan memprioritaskan tugas mereka.
- 3) Integrasi kehidupan kerja: Aplikasi ini bertujuan untuk membantu Anda mencapai keseimbangan kehidupan kerja dengan menyediakan alat untuk mengelola dan menggabungkan tugas dari kedua area.
- 4) Mengembangkan keterampilan manajemen waktu: Pengguna diharapkan dapat mengembangkan keterampilan manajemen waktu dengan menggunakan aplikasi ini, yang pada akhirnya dapat berdampak positif pada karir dan kehidupan pribadinya.

BAB II

LANDASAN TEORI

2.1 Pemrograman Berbasis Objek

Pemrograman Berbasis Objek Pemrograman Berbasis Objek (PBO) merupakan paradigma pemrograman yang menggunakan konsep objek sebagai unit dasar pengembangan program.

Beberapa konsep dasar yang digunakan dalam PBO adalah:

2.1.1 Konsep Kelas

Konsep Kelas Kelas adalah struktur dasar yang digunakan untuk mendefinisikan objek di PBO. Sebuah kelas mendeskripsikan atribut (data) dan metode (fungsi) yang dimiliki suatu objek. Misalnya, jika Anda membuat aplikasi daftar tugas, Anda bisa membuat kelas "Tugas" yang berisi atribut seperti nama tugas, status penyelesaian, dan tenggat waktu.

2.1.2 Konsep Objek

Konsep Objek Objek adalah turunan konkrit dari suatu kelas. Dalam konteks aplikasi daftar tugas, objek dapat mewakili tugas tertentu. Misalnya, objek pembelian bahan makanan adalah turunan dari kelas Task dengan atribut seperti status penyelesaian dan tenggat waktu.

2.1.3 Inheritance

Warisan Warisan memungkinkan Anda membuat kelas baru dengan mewarisi atribut dan metode kelas yang sudah ada.

Dalam konteks aplikasi daftar tugas, mungkin terdapat hierarki kelas seperti Tugas, yang mewarisi kelas DailyTask atau MinimumTask, dengan atribut tambahan bergantung pada jenis tugas.

2.1.4 Polimorfisme

Polimorfisme Polimorfisme memungkinkan objek memiliki metode dengan nama yang sama yang dapat diimplementasikan dengan cara berbeda.

Dalam aplikasi daftar tugas, Anda bisa menerapkan polimorfisme ke metode CompleteTask, yang memiliki implementasi berbeda tergantung pada tipe tugas.

2.2 Aplikasi To-Do List

2.2.1 Tinjauan Literatur

Tinjauan Pustaka Ada banyak aplikasi To-Do List yang sudah ada seperti Todolist, Microsoft To-Do, dan Any.do. Tinjauan literatur menunjukkan bahwa aplikasi ini umumnya menawarkan antarmuka yang intuitif, kemampuan untuk mengatur tugas berdasarkan prioritas, mengelompokkan tugas ke dalam kategori, dan berintegrasi dengan kalender dan notifikasi.

2.

2.2.2 Fitur-Fitur Umum

Beberapa fitur umum yang ditemui dalam aplikasi To-Do List meliputi:

- 1) Pengelompokan tugas:** Kemampuan mengelompokkan tugas berdasarkan kategori atau proyek.
- 2) Prioritas:** Penetapan tingkat prioritas untuk setiap tugas.
- 3) Notifikasi:** Notifikasi atau pengingat tentang tugas yang akan datang.
- 4) Integrasi kalender:** sinkronisasi dengan kalender Anda untuk melihat jadwal tugas.
- 5) Deskripsi tugas:** Memberikan ruang untuk informasi tambahan tentang tugas tersebut.

BAB III

PEMBAHASAN

```
1 import sys
2 from PyQt5.QtWidgets import QApplication, QMainWindow, QStackedWidget
3 from PyQt5 import QtWidgets, uic
4 from plyer import notification
5
6
7 Ui_MainWindow1, QMainWindowBase1 = uic.loadUiType("UI1.ui")
8
```

import sys: Import modul sys untuk berinteraksi dengan interpreter Python.

from PyQt5.QtWidgets import QApplication, QMainWindow, QStackedWidget: Import kelas-kelas yang dibutuhkan dari modul PyQt5 untuk membuat aplikasi GUI.

from PyQt5 import QtWidgets, uic: Import modul QtWidgets dan uic dari PyQt5.

from plyer import notification: Import modul notification dari plyer untuk menampilkan notifikasi.

Ui_MainWindow1, QMainWindowBase1 = uic.loadUiType("UI1.ui"): Memuat desain UI dari file "UI1.ui" menggunakan uic.loadUiType.

```
8
9 class TodoApp1(QMainWindow, Ui_MainWindow1):
10     def __init__(self, stacked_widget):
11         super().__init__()
12         self.setupUi(self)
13         self.setWindowTitle('To Do List')
14         self.setGeometry(100, 100, 600, 390)
15         self.stacked_widget = stacked_widget
16         self.done.clicked.connect(self.donefunc)
17         self.miss.clicked.connect(self.missfunc)
18         self.notyet.clicked.connect(self.notyetfunc)
19
20     def switch_to_screen2(self):
21         # Create instance TodoApp2 and add it to the stacked widget
22         screen2 = TodoApp2(self.stacked_widget)
23         self.stacked_widget.addWidget(screen2)
24         # Set current index to the index of screen2
25         self.stacked_widget.setCurrentIndex(self.stacked_widget.indexOf(screen2))
26
```

```
26
27     def donefunc(self):
28         notification.notify(
29             title='Task Done!',
30             message='You have completed a task.',
31             app_name='To Do List'
32         )
33     def missfunc(self):
```

```

26
27
28     def donefunc(self):
29         notification.notify(
30             title='Task Done!',
31             message='You have completed a task.',
32             app_name='To Do List'
33         )
34     def missfunc(self):
35         pass
36
37     def notyetfunc(self):
38         pass
39
40
41     def switch_to_screen1():
42         stacked_widget = QStackedWidget()
43         screen1 = TodoApp1(stacked_widget)
44         stacked_widget.addWidget(screen1)
45         stacked_widget.setCurrentIndex(stacked_widget.indexOf(screen1))

```

```

36     def notyetfunc(self):
37         pass
38
39
40
41     def switch_to_screen1():
42         stacked_widget = QStackedWidget()
43         screen1 = TodoApp1(stacked_widget)
44         stacked_widget.addWidget(screen1)
45         stacked_widget.setCurrentIndex(stacked_widget.indexOf(screen1))
46         return stacked_widget
47
48 if __name__ == '__main__':
49     app = QApplication(sys.argv)
50     window = QMainWindow()
51     stacked_widget = switch_to_screen1()
52     window.setCentralWidget(stacked_widget)
53     window.show()
54     sys.exit(app.exec_())
55

```

class TodoApp1(QMainWindow, Ui_MainWindow1): Mendefinisikan kelas TodoApp1 yang merupakan turunan dari QMainWindow dan menggunakan desain UI dari Ui_MainWindow1.

def __init__(self, stacked_widget): Konstruktor kelas TodoApp1 yang menerima objek stacked_widget sebagai parameter.

self.setupUi(self): Inisialisasi antarmuka pengguna menggunakan metode setupUi dari desain UI.

self.done.clicked.connect(self.donefunc): Menghubungkan tombol "done" ke metode donefunc.

def switch_to_screen2(self): Metode untuk beralih ke screen2 (TodoApp2).

def donefunc(self): Metode yang dipanggil saat tombol "done" ditekan, menampilkan notifikasi bahwa sebuah tugas telah selesai.

def missfunc(self): Metode yang akan diimplementasikan untuk tombol "miss".

def notyetfunc(self): Metode yang akan diimplementasikan untuk tombol "not yet".

def switch_to_screen1(): Fungsi untuk membuat dan mengembalikan objek QStackedWidget yang berisi screen1 (TodoApp1).

if __name__ == '__main__': Mengeksekusi blok kode di bawahnya jika script dijalankan langsung.

app = QApplication(sys.argv): Membuat instance dari aplikasi QApplication.

window = QMainWindow(): Membuat instance dari QMainWindow sebagai window utama.

stacked_widget = switch_to_screen1(): Membuat objek QStackedWidget yang berisi screen1.


window.setCentralWidget(stack_widget): Menetapkan stacked_widget sebagai widget pusat window.

window.show(): Menampilkan window.

sys.exit(app.exec_()): Mengakhiri aplikasi saat keluar.



```
1 import sys
2 from PyQt5 import QtWidgets, uic, QtCore
3 from PyQt5.QtWidgets import QApplication, QMainWindow, QListWidgetItem
4 from PyQt5.QtCore import QTimer
5 from plyer import notification
6 from screen1 import TodoApp1
7
8 import sqlite3
9
10
11 Ui_MainWindow, QMainWindowBase = uic.loadUiType("UI2.ui")
12
13 class TodoApp2(QMainWindow, Ui_MainWindow):
14     def __init__(self, stacked_widget):
15         super().__init__()
16         self.setupUi(self)
17         self.setWindowTitle('To Do List')
18         self.setGeometry(100, 100, 845, 390)
19         self.stacked_widget = stacked_widget
20         self.initUI()
21
22     def initUI(self):
23         self.calendarWidget.selectionChanged.connect(self.change)
24
```



```
22     def initUI(self):
23         self.calendarWidget.selectionChanged.connect(self.change)
24
25         # Tombol add
26         self.add.clicked.connect(self.updatetask)
27         self.time_edit.setTime(QTime.currentTime())
28         self.lineEdit.clear()
29
30         #tombol save change
31         self.save.clicked.connect(self.savechange)
32         #tombol delete
33         self.delete_2.clicked.connect(self.deletetask)
34
35         self.conn = sqlite3.connect('tasks.db')
36         self.cursor = self.conn.cursor()
37
38         self.cursor.execute('CREATE TABLE IF NOT EXISTS tasks
39 | | | | | | (task text, time text, date text)')
40
41         self.retrieve_tasks()
42
43     def savechange(self):
44
45         items = [self.listWidget.item(i).text() for i in range(self.listWidget.count())]
46
```

```

47 self.cursor.execute('DELETE FROM tasks')
48 self.conn.commit()
49
50 for item_text in items:
51     task, time = item_text.split(' - ')
52     date = self.calendarWidget.selectedDate().toPyDate().strftime('%d-%m-%Y')
53     self.cursor.execute('INSERT INTO tasks VALUES (?, ?, ?)', (task, time, date))
54     self.conn.commit()
55
56 self.close()
57
58 # Kembali ke window sebelumnya (misalnya, window utama)
59 self.previous_window = TodoApp1(self.stacked_widget)
60 self.previous_window.show()
61
62 def switch_to_screen1(self):
63     self.stacked_widget.setCurrentIndex(0)
64
65 def deletetask(self):
66     selected_items = self.listWidget.selectedItems()
67
68     for item in selected_items:
69         text = item.text()
70         task, time = text.split(' - ')
71

```

```

70 task, time = text.split(' - ')
71
72 self.cursor.execute('DELETE FROM tasks WHERE task=? AND time=?', (task, time))
73 self.conn.commit()
74
75 for item in selected_items:
76     row = self.listWidget.row(item)
77     self.listWidget.takeItem(row)
78
79 def retrieve_tasks(self):
80     self.cursor.execute('SELECT * FROM tasks')
81     rows = self.cursor.fetchall()
82     for row in rows:
83         task = f"{row[0]} - {row[1]}"
84         item = QListWidgetItem(task)
85         item.setFlags(item.flags() | QtCore.Qt.ItemIsUserCheckable)
86         item.setCheckState(QtCore.Qt.Unchecked)
87         self.listWidget.addItem(item)
88
89 def updatetask(self):
90     if self.lineEdit.text():
91         task = self.lineEdit.text()
92         time = self.time_edit.time().toString("hh:mm")
93

```

```

105 self.listWidget.clear()
106 self.cursor.execute('SELECT * FROM tasks WHERE date=?', (selected_date,))
107 rows = self.cursor.fetchall()
108 for row in rows:
109     task = f"{row[0]} - {row[1]}"
110     item = QListWidgetItem(task)
111     item.setFlags(item.flags() | QtCore.Qt.ItemIsUserCheckable)
112     item.setCheckState(QtCore.Qt.Unchecked)
113     self.listWidget.addItem(item)
114
115 def showNotification(self, title, message):
116     notification.notify(
117         title=title,
118         message=message,
119         timeout=10
120     )
121
122 if __name__ == '__main__':
123     app = QApplication(sys.argv)
124     widget = QtWidgets.QStackedWidget()
125     todo_app = TodoApp2(widget)
126     todo_app.show()
127     sys.exit(app.exec_())
128

```

Imports: Sama seperti di kode pertama, namun juga mengimpor TodoApp1 dari screen1.

Ui_MainWindow, QMainWindowBase = uic.loadUiType("UI2.ui"): Memuat desain UI dari file "UI2.ui".

class TodoApp2(QMainWindow, Ui_MainWindow):: Mendefinisikan kelas TodoApp2 yang merupakan turunan dari QMainWindow dan menggunakan desain UI dari Ui_MainWindow.

def __init__(self, stacked_widget):: Konstruktor kelas TodoApp2 yang menerima objek stacked_widget sebagai parameter.

def initUI(self):: Inisialisasi antarmuka pengguna dan fungsionalitas lainnya.

def savechange(self):: Metode untuk menyimpan perubahan pada tugas ke database dan kembali ke screen sebelumnya (TodoApp1).

def switch_to_screen1(self):: Metode untuk beralih ke screen1 (TodoApp1).

def deletetask(self):: Metode untuk menghapus tugas yang dipilih.

def retrieve_tasks(self):: Metode untuk mengambil tugas dari database dan menampilkannya di listWidget.

def updatetask(self):: Metode untuk menambahkan tugas baru ke listWidget dan menyimpannya ke database.

def change(self):: Metode yang dipanggil saat terjadi perubahan pada calendarWidget, mengambil dan menampilkan tugas berdasarkan tanggal.

def showNotification(self, title, message):: Metode untuk menampilkan notifikasi menggunakan plyer.

if __name__ == '__main__': Mengeksekusi blok kode di bawahnya jika script dijalankan langsung.

app = QApplication(sys.argv): Membuat instance dari aplikasi QApplication.

widget = QtWidgets.QStackedWidget(): Membuat objek QStackedWidget sebagai wadah untuk screen2 (TodoApp2).

todo_app = TodoApp2(widget): Membuat instance dari TodoApp2 dengan menyertakan objek QStackedWidget sebagai parameter.

todo_app.show(): Menampilkan screen2.

sys.exit(app.exec_()): Mengakhiri aplikasi saat keluar.

BAB IV

PENUTUP

4.1 KESIMPULAN

Makalah ini membahas penerapan pemrograman berorientasi objek dalam pembuatan aplikasi To-Do List menggunakan PyQt5. Aplikasi ini memiliki dua antarmuka pengguna yang memungkinkan pengguna mengelola tugas-tugas mereka dengan cara yang interaktif dan mudah dipahami. Pemrograman berorientasi objek digunakan untuk merancang kelas-kelas yang memodelkan entitas seperti jendela aplikasi, antarmuka pengguna, dan fungsi-fungsi terkait. Penerapan konsep seperti inheritance, encapsulation, dan polymorphism dapat ditemui dalam struktur kelas, memungkinkan pengelolaan tugas dan navigasi antarmuka secara efisien. Qt Designer digunakan untuk merancang antarmuka pengguna, memisahkan desain dari logika aplikasi untuk memudahkan pemeliharaan dan pengembangan.

Aplikasi ini memanfaatkan fitur-fitur PyQt5 seperti QMainWindow, QListWidget, QCalendarWidget, dan plyer untuk notifikasi. Selain itu, penggunaan modul plyer memungkinkan aplikasi untuk memberikan notifikasi ke pengguna di sistem operasi yang berbeda.

4.2 SARAN

Menyadari bahwa tidak ada yang sempurna selain Yang Maha Kuasa, Makalah ini juga belum sempurna karena keterbatasan pengetahuan kami. Kedepannya kami akan senantiasa giat belajar mengenai Pemrograman Berorientasi Objek untuk meningkatkan kapasitas pada mata kuliah kedepannya, diharapkan mata kuliah pemrograman berorientasi objek dapat memberikan dasar yang kokoh dan mempersiapkan mahasiswa untuk menghadapi tantangan dalam pengembangan perangkat lunak berbasis objek di dunia nyata.

DAFTAR PUSTAKA

(Covey, S. R. (1989). The 7 Habits of Highly Effective People: Powerful Lessons in Personal Change. Free Press.)

Allen, D. (2001). Getting Things Done: The Art of Stress-Free Productivity. Penguin.

Allen, D. (2001). Getting Things Done: The Art of Stress-Free Productivity. Penguin.

Covey, S. R. (1989). The 7 Habits of Highly Effective People: Powerful Lessons in Personal Change. Free Press.

Shariati, F., & Hatamian, M. (2017). A Comparative Analysis of Android To-Do List Apps Based on User Preferences. International Journal of Engineering Research and Applications, 7(4), 46-53.

Gartner. (2022). Magic Quadrant for Personal and Team Productivity. Gartner, Inc.