Курсова робота на тему:

**«Аналіз успішності студентів учбового закладу»**

**Зміст**

Зміст……………………………………………………………………………….......2

Вступ………………………………………………………………………………......3

1. Огляд MVC

1.1 Загальна характеристика

1.2 Структура MVC

1.3 Шаблони програмування, що використовуються в MVC

1.4 Завдання MVC

1.5 Python

1.6 Використання концепції MVC в даній роботі

2. Проектування програмного додатку……………………………………..………x

**Вступ**

Об’єктом розробки даної курсової роботи є програма з функціональністю аналізу успішності студентів учбового закладу.

Метою курсової роботи є закріплення теоретичних знань і практичних навичок з проектування, моделювання, розробки та тестування програмного забезпечення з графічним інтерфейсом.

Результат курсової роботи – готовий програмний додаток з графічним інтерфейсом, написаний на мові програмування Python.

Графічний інтерфейс користувача – інтерфейс між комп’ютером та його користувачем, реалізований за допомогою фреймворка PyQt.

**Огляд MVC**

1. *Загальна характеристика*

Model – view – controller (MVC, «модель – представлення – поведінка») – схема використання кількох шаблонів проектування, за допомогою яких модель даних програми, інтерфейс користувача і взаємодія з користувачем розділені на три окремих компонента таким чином, щоб модифікація одного з компонентів впливала на інші мінімально. Дана схема проектування часто використовується для побудови архітектурного каркаса, коли переходять від теорії до реалізації в конкретній предметній області.

Мета шаблону — гнучкий дизайн програмного забезпечення, який повинен полегшувати подальші зміни чи розширення програм, а також надавати можливість повторного використання окремих компонентів програми. Крім того використання цього шаблону у великих системах сприяє впорядкованості їхньої структури і робить їх більш зрозумілими за рахунок зменшення складності.

1. *Структура MVC*

У рамках архітектурного шаблону модель–вигляд–контролер (MVC) програма поділяється на три окремі, але взаємопов'язані частини з розподілом функцій між компонентами. Модель (Model) відповідає за зберігання даних і забезпечення інтерфейсу до них. Вигляд (View) відповідальний за представлення цих даних користувачеві. Контролер (Controller) керує компонентами, отримує сигнали у вигляді реакції на дії користувача (зміна положення курсора миші, натискання кнопки, ввід даних в текстове поле) і передає дані у модель.

* Модель (Model) є центральним компонентом шаблону MVC і відображає поведінку застосунку, незалежну від інтерфейсу користувача. Модель стосується прямого керування даними, логікою та правилами застосунку.
* Вигляд (View) може являти собою будь-яке представлення інформації, одержуване на виході, наприклад графік чи діаграму. Одночасно можуть співіснувати кілька виглядів (представлень) однієї і тієї ж інформації, наприклад гістограма для керівництва компанії й таблиці для бухгалтерії.
* Контролер (Controller) одержує вхідні дані й перетворює їх на команди для моделі чи вигляду.

Модель інкапсулює ядро даних і основний функціонал їхньої обробки і не залежить від процесу вводу чи виводу даних. Вигляд може мати декілька взаємопов'язаних областей, наприклад різні таблиці і поля форм, в яких відображаються дані.

У функції контролера входить відстеження визначених подій, що виникають в результаті дій користувача. Контролер дозволяє структурувати код шляхом групування пов'язаних дій в окремий клас. Наприклад у типовому MVC-проекті може бути користувацький контролер, що містить групу методів, пов'язаних з управлінням обліковим записом користувача, таких як реєстрація, авторизація, редагування профілю та зміна пароля.

1. *Шаблони програмування, що використовуються в MVC*

Для реалізації схеми MVC використовується досить велика кількість шаблонів проектування (залежно від складності архітектурного рішення), основні з яких «спостерігач», «стратегія» та «компонувальник».

Найбільш типова реалізація відокремлює представлення від моделі шляхом встановлення між ними протоколу взаємодії, використовуючи апарат подій (підписка/сповіщення). При кожній зміні внутрішніх даних в моделі вона оповіщає все залежні від неї представлення, і вони оновлюються. Для цього використовується шаблон «спостерігач» При обробці реакції користувача представлення вибирає, залежно від потрібної реакції, потрібний контролер, який забезпечить той чи інший зв’язок з моделлю. Для цього використовується шаблон «стратегія», або замість цього може бути модифікація з використанням шаблону «команда». Крім того, можуть використовуватися й інші шаблони проектування, наприклад, «фабричний метод», який дозволить задати за замовчуванням тип контролера для відповідного представлення.

1. *Завдання MVC*

Основна мета застосування цієї концепції полягає в відокремленні бізнес-логіки (моделі) від її візуалізації (представлення, виду). За рахунок такого поділу підвищується можливість повторного використання. Найбільш корисне застосування даної концепції в таких випадках, коли користувач повинен бачити ті ж самі дані одночасно в різних контекстах та/або з різних точок зору. Зокрема, виконуються наступні завдання:

* До однієї моделі можна приєднати кілька представлень, при цьому не зачіпаючи реалізацію моделі. Наприклад, деякі дані можуть бути одночасно представлені у вигляді електронної таблиці, гістограми і діаграми.
* Не торкаючись реалізації представлень, можна змінити реакції на дії користувача (натискання на кнопку, введення даних), для цього досить використовувати інший контролер.
* Ряд розробників спеціалізується тільки в одній з областей: або розробляють графічний інтерфейс, або розробляють бізнес-логіку. Тому можна досягти того, що програміст, які займаються розробкою бізнес-логіки (моделі), взагалі не будуть знати про те, який вигляд буде використовуватися.

1. *Python*

Дану курсову роботу буде виконано, використовуючи мову програмування Python та бібліотеку PyQt. Бібліотека PyQT зручна та має велику кількість можливостей. Вона зможе надати нам змогу управляти графічними виглядом елементів.

1. *Використання концепції MVC в даній роботі*

В даній роботі буде використовуватися шаблон MVC, який розділить систему на три частини, а саме: модель даних (база даних або файл), вигляд даних (графічний інтерфейс користувача) та керування (логіка програми). Це буде зроблено з метою відокремлення моделі даних від інтерфейсу користувача, щоб зміни в будь-якій з цих частин системи мінімально впливали на інші частину нашої системи.

**Проектування програмного додатку**