Міністерство освіти і науки України
Національний технічний університет України “Київський політехнічний інститут імені Ігоря Сікорського”
Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем та технологій

Лабораторна робота №2
З дисципліни «Технології розроблення програмного забезпечення»
Тема: **«ДІАГРАМА ВАРІАНТІВ ВИКОРИСТАННЯ. СЦЕНАРІЇ ВАРІАНТІВ ВИКОРИСТАННЯ. ДІАГРАМИ UML. ДІАГРАМИ КЛАСІВ. КОНЦЕПТУАЛЬНА МОДЕЛЬ СИСТЕМИ»**
Flex Automatical tool

Виконав:
Студент групи ІА-22
Сидорін Д.О.

Перевірив:
Мягкий М. Ю.

Київ-2025

Зміст

Тема:

Діаграма варіантів використання. Сценарії варіантів використання. Діаграми uml. Діаграми класів. Концептуальна модель системи

Мета:

Дослідити та застосувати UML-діаграми для моделювання варіантів використання і концептуальної структури даних системи, зосереджуючись на діаграмах класів, прецедентів та опису їхніх сценаріїв використання.

Завдання:

Розробити візуальний додаток для автоматизації, що включає функції для створення та управління кількома автоматичними завданнями (у вкладках). Користувач повинен мати можливість автоматично налаштовувати і виконувати дії, такі як завантаження нових файлів, оновлення статусів в месенджерах, виконання макросів. У додатку має бути можливість додавати категорії для завдань (терміновість, типи дій) та автоматично виконувати їх згідно з заданими правилами. Також повинна бути можливість візуалізації завдань у вигляді "карт пам'яті", де кожне завдання або макрос буде представлено у вигляді вузла з можливістю додавати вкладені файли, зображення, відео (з попереднім переглядом).

Хід роботи

1. Схема прецедентів

2. Деталі сценаріїв використання

1. Вхід у систему

Основний сценарій:

1. Користувач запускає додаток.
2. Система запитує логін і пароль.
3. Користувач вводить правильні дані.
4. Система перевіряє введені дані.
5. Якщо дані правильні, система надає доступ до інтерфейсу управління автоматизацією.
6. Якщо дані неправильні, система виводить повідомлення про помилку.

Альтернативний сценарій:

- 3а. Якщо користувач забув пароль:
    1. Користувач натискає кнопку "Забули пароль?".
    2. Система відправляє інструкції на електронну пошту користувача.

---

2. Створення правила автоматизації

Основний сценарій:

1. Користувач вибирає опцію "Створити нове правило".
2. Система відображає форму для введення параметрів правила.
3. Користувач вводить:
    - Тип тригера (наприклад, час, активність користувача).
    - Умови для тригера (наприклад, "щоп'ятниці о 20:00").
    - Тип дії (наприклад, завантажити файл, змінити статус).
    - Параметри дії (наприклад, URL для завантаження файлів або конкретний статус для комунікатора).
4. Користувач натискає "Зберегти".
5. Система зберігає правило і відображає підтвердження.

Альтернативний сценарій:

- 4а. Якщо введено некоректні дані (наприклад, неправильний URL):
    1. Система виводить повідомлення про помилку і запитує виправлення.

---

3. Редагування правила

Основний сценарій:

1. Користувач вибирає існуюче правило.

2. Система завантажує дані цього правила.
3. Користувач змінює необхідні параметри.
4. Користувач натискає "Зберегти зміни".
5. Система зберігає зміни і підтверджує успіх.

Альтернативний сценарій:

- 4а. Якщо зміни не можуть бути збережені (наприклад, помилка з'єднання з базою даних):
    1. Система виводить повідомлення про помилку.

---

4. Видалення правила

Основний сценарій:

1. Користувач вибирає правило для видалення.
2. Система запитує підтвердження на видалення.
3. Користувач підтверджує видалення.
4. Система видаляє правило з бази даних і підтверджує успіх.

Альтернативний сценарій:

- 3а. Якщо користувач скасовує видалення:
    1. Система повертає користувача до списку правил без змін.

---

5. Виконання макросів

Основний сценарій:

1. Користувач вибирає опцію "Запустити макрос".
2. Система перевіряє, чи є налаштовані макроси для виконання.
3. Система автоматично виконує всі макроси, які відповідають умовам тригера.
4. Система надає користувачу звіт про виконання макросів.

Альтернативний сценарій:

- 3а. Якщо немає доступних макросів для виконання:
    1. Система виводить повідомлення, що жоден макрос не був знайдений.

---

6. Завантаження файлів

Основний сценарій:

1. Користувач налаштовує тригер для автоматичного завантаження файлів (наприклад, щоп'ятниці о 20:00).
2. Система перевіряє, чи є нові файли для завантаження.

3. Якщо файли є, система завантажує їх з попередньо вказаного джерела.
4. Система відображає повідомлення про успішне завершення завантаження.

Альтернативний сценарій:

- 3а. Якщо файли не можуть бути завантажені через помилку з'єднання:
    1. Система виводить повідомлення про помилку і надає можливість повторити спробу.

---

7. Зміна статусу комунікатора

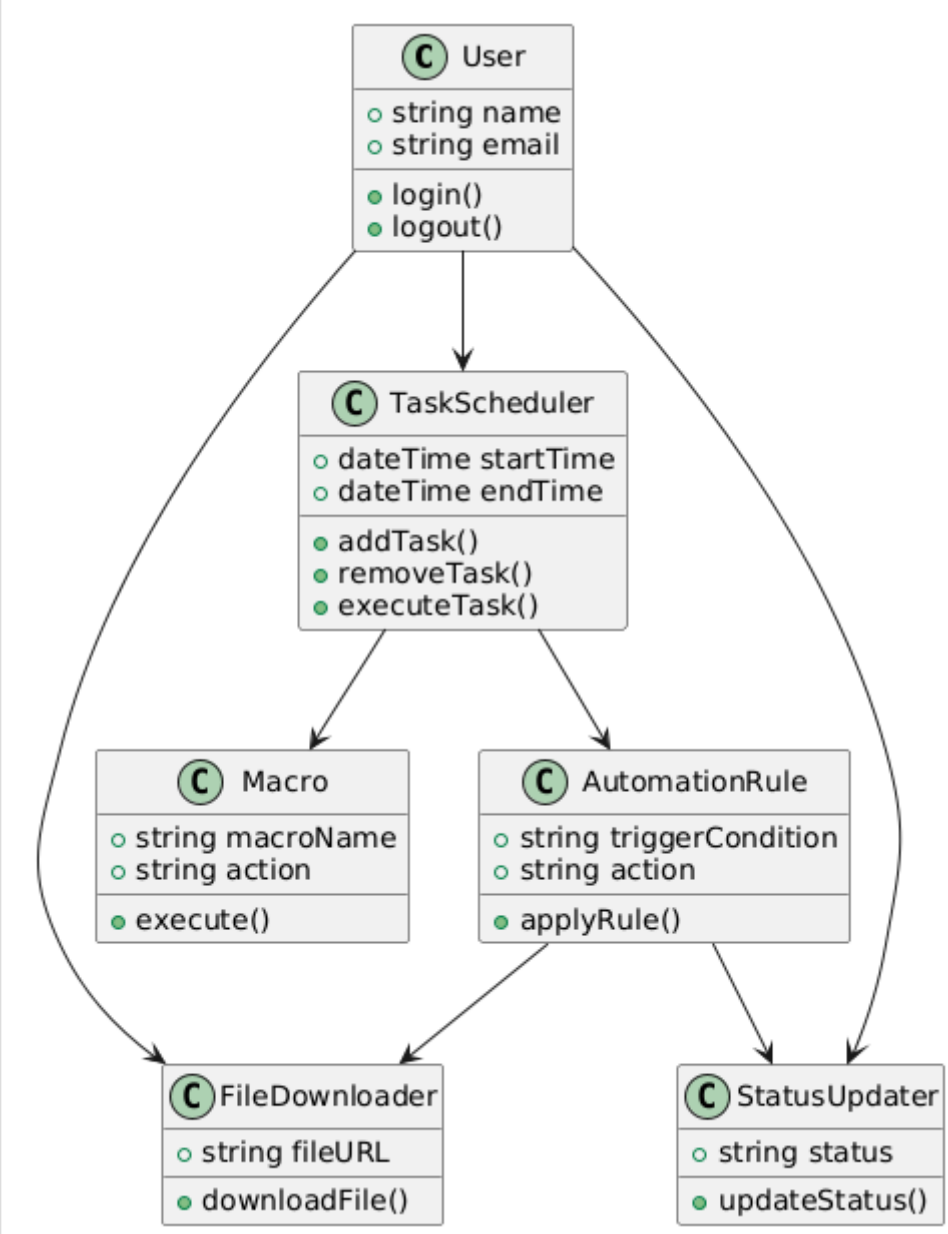
Основний сценарій:

1. Користувач налаштовує тригер для зміни статусу комунікатора (наприклад, якщо неактивність понад 10 хвилин).
2. Система перевіряє активність користувача.
3. Якщо активність відсутня більше 10 хвилин, система змінює статус комунікатора на "Відійшов".
4. Система відображає підтвердження зміни статусу.

Альтернативний сценарій:

- 3а. Якщо комунікатор не підтримує зміну статусу через програмний інтерфейс:
    1. Система виводить повідомлення про помилку та пропонує інші варіанти дій.

3. Схема класів



Ця діаграма класів описує систему автоматизації, яка забезпечує зручність для користувачів шляхом виконання автоматичних дій, таких як завантаження файлів, оновлення статусів та виконання макросів. Ось детальний опис основних класів і їхніх взаємозв'язків:

1. User

- **Опис**: Клас, який представляє користувача системи. Користувач взаємодіє із системою, налаштовує автоматизацію та виконувані завдання.
- **Атрибути**:
  - name: ім'я користувача.
  - email: електронна пошта користувача.
- **Методи**:

- login(): метод для входу користувача в систему.
- logout(): метод для виходу з системи.

Взаємозв'язки: Клас **User** асоціюється з **TaskScheduler**, **StatusUpdater** та **FileDownloader**, оскільки користувач може ініціювати завдання, оновлювати статуси або завантажувати файли.

2. TaskScheduler

- **Опис**: Клас для планування завдань, що виконуються у визначений час або згідно з заданими правилами.
- **Атрибути**:
  - startTime: час початку виконання завдання.
  - endTime: час завершення виконання завдання.
- **Методи**:
  - addTask(): додавання нового завдання.
  - removeTask(): видалення існуючого завдання.
  - executeTask(): виконання запланованого завдання.

Взаємозв'язки: **TaskScheduler** має зв'язок з **Macro** та **AutomationRule** для виконання макросів і правил автоматизації. Це дозволяє планувати завдання на конкретний час чи під певні умови.

3. Macro

- **Опис**: Клас для опису макросів, які автоматизують серії дій, таких як натискання клавіші чи рух миші.
- **Атрибути**:
  - macroName: ім'я макросу.
  - action: опис дії, яку виконує макрос.
- **Методи**:
  - execute(): виконання макросу.

Взаємозв'язки: **Macro** використовується в **TaskScheduler** для виконання автоматичних дій, таких як натискання клавіші або рух миші.

4. FileDownloader

- **Опис**: Клас для завантаження файлів з Інтернету, таких як фільми або книги.
- **Атрибути**:
  - fileURL: URL файлу, який потрібно завантажити.
- **Методи**:
  - downloadFile(): метод для завантаження файлу.

Взаємозв'язки: **FileDownloader** використовується в **AutomationRule** для автоматичного завантаження файлів, таких як нові епізоди серіалів або книги.

5. StatusUpdater

- **Опис**: Клас для оновлення статусу користувача в комунікаторах, таких як Skype.
- **Атрибути**:
  - status: поточний статус користувача (наприклад, "Доступний", "Не турбувати").
- **Методи**:
  - updateStatus(): метод для оновлення статусу.

Взаємозв'язки: **StatusUpdater** може бути використаний у **AutomationRule** для автоматичного оновлення статусу користувача за певними умовами.

6. AutomationRule

- **Опис**: Клас, який описує правило автоматизації, яке складається з умови та відповідної дії.
- **Атрибути**:
  - triggerCondition: умова для активації правила (наприклад, відсутність активності).
  - action: дія, яка виконується після спрацьовування умови (наприклад, оновлення статусу або завантаження файлу).
- **Методи**:
  - applyRule(): метод для застосування правила.

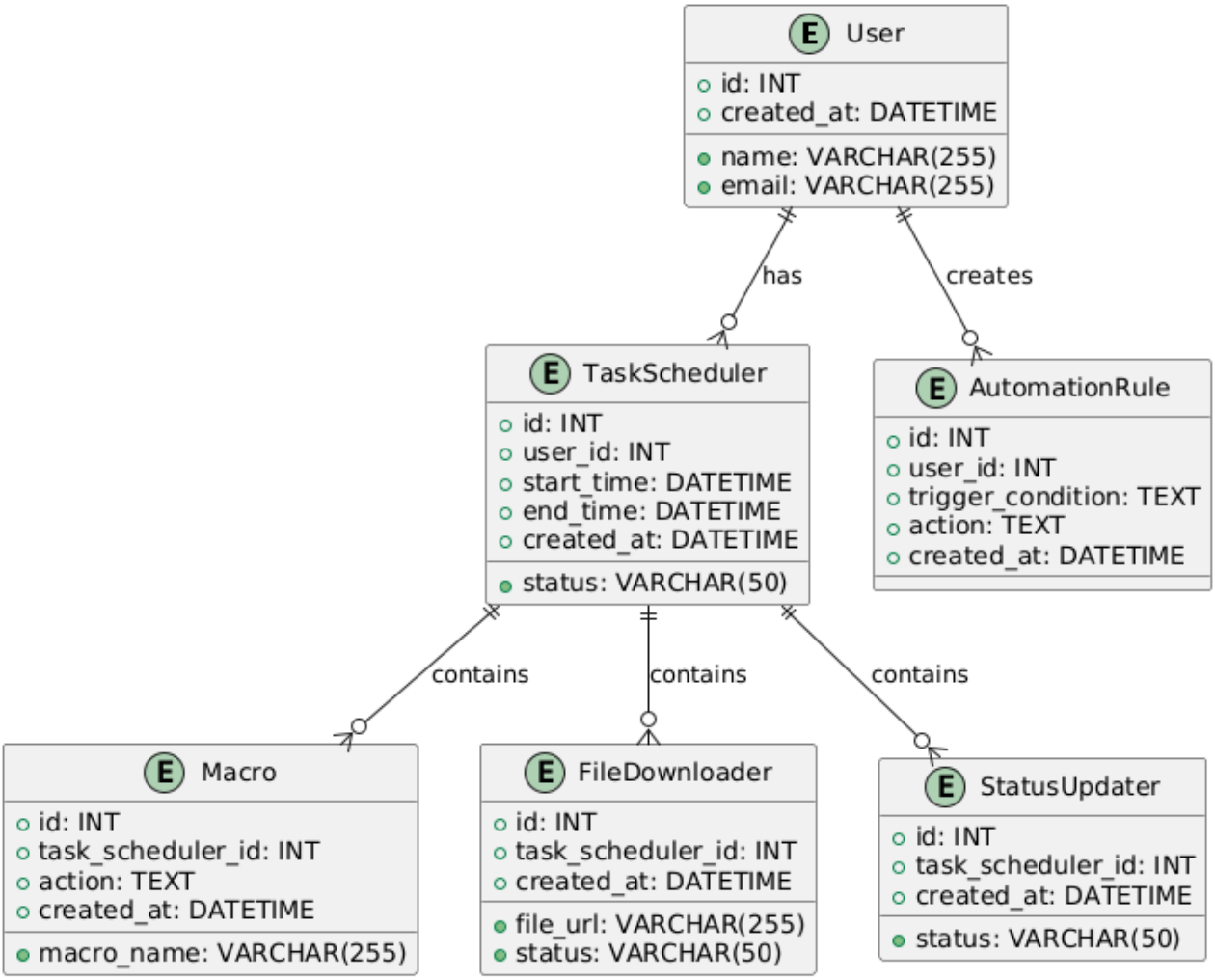
Взаємозв'язки: **AutomationRule** взаємодіє з **StatusUpdater** та **FileDownloader** для виконання дій, таких як оновлення статусу або завантаження файлів, на основі заданих умов.

Основні взаємозв'язки:

- **User** має зв'язок з **TaskScheduler**, **StatusUpdater**, **FileDownloader** через взаємодію з ними.
- **TaskScheduler** взаємодіє з **Macro** та **AutomationRule** для планування та виконання автоматичних завдань.
- **AutomationRule** пов'язує **StatusUpdater** та **FileDownloader** для виконання конкретних автоматичних дій.

Ця діаграма класів показує основні компоненти вашої системи автоматизації та їхні взаємозв'язки для реалізації завдань, таких як завантаження файлів, оновлення статусів та виконання макросів.

4. Структура бази даних



Опис таблиць і їхніх зв'язків:

1. **User**: таблиця користувачів, яка зберігає дані користувачів, що використовують систему.
    - **Атрибути**:
        - id: унікальний ідентифікатор користувача.
        - name: ім'я користувача.
        - email: електронна пошта користувача.
        - created_at: дата та час створення запису.
2. **TaskScheduler**: таблиця для зберігання інформації про заплановані завдання.
    - **Атрибути**:
        - id: унікальний ідентифікатор завдання.
        - user_id: зв'язок з таблицею **User**, ідентифікатор користувача, який створив завдання.
        - start_time: час початку завдання.

        - end_time: час закінчення завдання.
        - status: статус завдання (наприклад, "активне", "завершено").
        - created_at: дата та час створення завдання.
3. **Macro**: таблиця для зберігання макросів, які автоматизують дії користувача.
    - **Атрибути**:
        - id: унікальний ідентифікатор макросу.
        - task_scheduler_id: зв'язок з таблицею **TaskScheduler**, ідентифікатор завдання, до якого відноситься макрос.
        - macro_name: ім'я макросу.
        - action: дія, яку виконує макрос.
        - created_at: дата та час створення макросу.
4. **FileDownloader**: таблиця для зберігання інформації про файли, які завантажуються через систему.
    - **Атрибути**:
        - id: унікальний ідентифікатор завантаження.
        - task_scheduler_id: зв'язок з таблицею **TaskScheduler**, ідентифікатор завдання, до якого відноситься завантаження файлів.
        - file_url: URL файлу, який потрібно завантажити.
        - status: статус завантаження (наприклад, "в процесі", "завершено").
        - created_at: дата та час створення запису про завантаження.
5. **StatusUpdater**: таблиця для зберігання статусів, які оновлюються в системі.
    - **Атрибути**:
        - id: унікальний ідентифікатор статусу.
        - task_scheduler_id: зв'язок з таблицею **TaskScheduler**, ідентифікатор завдання, до якого відноситься оновлення статусу.
        - status: новий статус користувача (наприклад, "Доступний", "Не турбувати").
        - created_at: дата та час зміни статусу.
6. **AutomationRule**: таблиця для зберігання правил автоматизації, які користувач може налаштувати для виконання дій.
    - **Атрибути**:
        - id: унікальний ідентифікатор правила.
        - user_id: зв'язок з таблицею **User**, ідентифікатор користувача, який створив правило.

- trigger_condition: умова, яка повинна бути виконана для активації правила.
- action: дія, яка виконується після спрацьовування правила.
- created_at: дата та час створення правила.

Зв'язки:

- **User** має зв'язок "один до багатьох" з **TaskScheduler** та **AutomationRule**, оскільки один користувач може створювати багато завдань та правил.
- **TaskScheduler** має зв'язок "один до багатьох" з **Macro**, **FileDownloader**, **StatusUpdater**, оскільки одне завдання може мати кілька макросів, завантажень файлів та оновлень статусів.

Висновки:

Дослідили та застосували UML-діаграми для моделювання варіантів використання і концептуальної структури даних системи, зосереджуючись на діаграмах класів, прецедентів та опису їхніх сценаріїв використання.