Зміст

[1 Огляд MVC 3](#_Toc383433322)

[1.1 Загальний огляд 3](#_Toc383433323)

[1.2 Історія 4](#_Toc383433324)

[1.3 Призначення 5](#_Toc383433325)

[1.4 Концепція 5](#_Toc383433326)

[1.5 Найчастіші помилки 7](#_Toc383433327)

[1.6 Реалізація 8](#_Toc383433328)

[1.6.1 Java 8](#_Toc383433329)

[2 Проектування програмного забезпечення 9](#_Toc383433330)

[2.1 Use case 9](#_Toc383433331)

[2.1.1 Реєстрація нового користувача 9](#_Toc383433332)

[2.1.2 Управління профілем 11](#_Toc383433333)

[2.1.3 Замовлення довідки 11](#_Toc383433334)

[2.1.4 Залишити повідомлення для викладача/ групи 12](#_Toc383433335)

[2.1.5 Пергляд розкладу студента/викладача 13](#_Toc383433336)

[2.1.6 Авторизація у системі 13](#_Toc383433337)

[2.2 Проектування графічного інтерфейсу користувача 14](#_Toc383433338)

[2.2.1 Макет графічного інтерфейсу 14](#_Toc383433339)

[2.3 Діаграма граничних класів 16](#_Toc383433340)

# Огляд MVC

## Загальний огляд

**Модель-вид-контролер** (або **Модель-вигляд-контролер**,  **Модель-представлення-контролер** [англ.](http://uk.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D1%96%D0%B9%D1%81%D1%8C%D0%BA%D0%B0_%D0%BC%D0%BE%D0%B2%D0%B0) *Model-view-controller*, **MVC**) — [архітектурний шаблон](http://uk.wikipedia.org/wiki/%D0%90%D1%80%D1%85%D1%96%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%BD%D1%96_%D1%88%D0%B0%D0%B1%D0%BB%D0%BE%D0%BD%D0%B8_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE_%D0%B7%D0%B0%D0%B1%D0%B5%D0%B7%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%BD%D1%8F), який використовується під час проектування та розробки [програмного забезпечення](http://uk.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BD%D0%B5_%D0%B7%D0%B0%D0%B1%D0%B5%D0%B7%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%BD%D1%8F).



Рис 1.1 Концепція Model-View-Controller

Цей шаблон поділяє систему на три частини: [модель даних](http://uk.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C_%D0%B4%D0%B0%D0%BD%D0%B8%D1%85), вигляд даних та [керування](http://uk.wikipedia.org/wiki/%D0%9A%D0%B5%D1%80%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F). Застосовується для відокремлення даних (модель) від [інтерфейсу користувача](http://uk.wikipedia.org/wiki/%D0%86%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81_%D0%BA%D0%BE%D1%80%D0%B8%D1%81%D1%82%D1%83%D0%B2%D0%B0%D1%87%D0%B0)(вигляду) так, щоб зміни інтерфейсу користувача мінімально впливали на роботу з даними, а зміни в моделі даних могли здійснюватися без змін інтерфейсу користувача.

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 призводить до певної впорядкованості їх структури і робить їх зрозумілішими завдяки зменшенню складності.

Дана схема проектування часто використовується для побудови архітектурного каркасу, коли переходять від теорії до реалізації в конкретній предметній області.

## Історія

Концепція MVC була описана у 1979 році Трюгве Реенскауг (англ. Trygve Reenskaug), який тоді працював над мовою програмування Smalltalk в Xerox PARC. Оригінальна реалізація описана в статті «Applications Programming in Smalltalk-80: How to use Model-View-Controller». Потім Джим Алтоф з командою розробників реалізували версію MVC для бібліотеки класів Smalltalk-80.

В оригінальній концепції була описана сама ідея і роль кожного із елементів: моделі, представлення і контролера. Та зв’язки між ними були описані без конкретизації. Крім того, розрізняли дві основні модифікації:

1. Пасивна модель – модель не має ніяких способів впливу на представлення чи контролер і використовується ними в якості джерела даних для відображення. Всі зміни моделі відстежуються контролером і він же відповідає за перемальовування представлення, якщо це необхідно. Така модель частіше всього використовується в структурному програмуванні, оскільки у цьому випадку модель являє собою просто структуру даних, без методів їх обробки.
2. Активна модель – модель сповіщає представлення про те, що в ній виникли зміни, а представлення, які зацікавленні в сповіщенні, підписуються на ці сповіщення. Це дозволяє зберегти незалежність моделі як від контролера, так і від представлення.

Класичною реалізацією концепції MVC вважають версію з активною моделлю.

Із розвитком Об’єктно-орієнтованого програмування і поняття про шаблони проектування був створений ряд модифікацій концепції MVC, які при реалізації у різних авторів можуть відрізнятися від оригінальної. Так, наприклад, Еріан Вермі в 2004 році писав приклад узагальненого MVC.

## Призначення

Основна мета використання цієї концепції у розділенні бізнес-логіки(моделі) від її візуалізації(представлення, вигляду). За рахунок такого розмежування підвищується можливість повторного використання. Найбільш корисне застосування даної теорії у тих випадках, коли користувач повинен бачити ті ж самі дані одночасно у різних контекстах та/або з різних точок зору. Виконуються наступні задачі:

1. До однієї моделі можна приєднати декілька видів, при цьому на чіпаючи реалізацію моделі. Наприклад, деякі дані можуть бути одночасно представлені у вигляді електронної таблиці, гістограми і кругової діаграми.
2. Не чіпаючи реалізацію вигляду, можна змінити реакції на дії користувача(натиснення мишкою на копку, введення даних), для цього достатньо використовувати інший контролер.
3. Ряд розробників спеціалізуються лише в одній із областей: або розробляють графічний інтерфейс, або розробляють бізнес-логіку. Тому можливо досягнути того, що програмісти, які займаються розробкою бізнес-логіки(моделі), взагалі не будуть обізнані в тому, яке представлення використовуватиметься.

## Концепція

Концепція MVC дозволяє розділити дані, представлення і обробку дій користувача на три окремих компоненти:

1. Модель(англ. Model) Модель представляє інформацію: дані і методи роботи з цими даними, реагує на запити, змінюючи свій стан. Не містить інформації, про те, як ці знання можна візуалізувати.
2. Представлення, вигляд (англ. View). Відповідає за відображення інформації(візуалізацію). Часто в якості представлення виступає форма(вікно) з графічними елементами.
3. Контролер (англ. Controller). Забезпечує зв’язок між користувачем і системою: контролює введення даних користувачем і використовує модель і представлення для реалізації необхідної реакції, керує компонентами, отримує сигнали у вигляді реакції на дії користувача, і повідомляє про зміни компоненту Модель.

Така внутрішня структура в цілому поділяє систему на самостійні частини і розподіляє відповідальність між різними компонентами.

Важливо зазначити, що як представлення, так і контролер залежать від моделі. Однак модель не залежить ні від представлення, ні від контролера. Тим самим досягається призначення такого розмежування: воно дозволяє будувати модель незалежно від візуального представлення, а також створювати декілька різних представлень для однієї моделі

Для реалізації схеми Model-View-Controller використовується достатньо велике число шаблонів проектування(в залежності від складності архітектурного рішення), основні з яких стратегія, композит, спостерігач.

Найбільш типова реалізація відділяє вигляд від моделі шляхом встановлення між ними протоколу взаємодії, використовуючи апарат подій(підписка/сповіщення). При кожній зміні внутрішніх даних в моделі вона сповіщає всі залежні від неї представлення і представлення обновлюється. Для цього використовують шаблон спостерігач. При обробці реакції користувача вигляд обирає, в залежності від потрібної реакції, потрібний контролер, який забезпечує той чи інший зв’язок з моделлю. Для цього використовується шаблон стратегія, або замість цього може бути модифікація із використанням шаблону команда. А для можливості однотипної роботи з підоб’єктами складно-скомпонованого ієрархічного виду може використовуватись шаблон композит. Крім того, можуть використовуватись і інші шаблони проектування, наприклад, фабричний метод, який дозволить задати за замовчуванням тип контролера для відповідного виду.

## Найчастіші помилки

Програмісти-початківці (особливо в веб-програмуванні, де абревіатура MVC стала популярною) дуже часто трактують архітектурну модель MVC як пасивну модель MVC. У цьому випадку модель виступає виключно сукупністю функцій для доступу до даних, а контролер містить бізнес логіку. В результаті код моделі по факту являється засобом для отримання даних із СУБД, а контролер являє собою типовий модуль, наповнений бізнес-логікою, або скрипт у термінології веб-програміста. В результаті такого розуміння MVC розробники стали писати код, який Pádraic Brady, відомий у колах спільноти Zend Framework, охарактеризував як ТТПК – «Товсті тупі потворні контролери» (Fat Stupid Ugly Controllers)

Середньостатистичний ТТПК отримував дані із БД (використоруючи рівень абстракції бази даних, враховуючи, що ще модель) або маніпулював, перевіряв, записував, а також передавав дані у вигляд. Такий підхід став дуже популярним тому, що використання таких контролерів схоже на класичну практику використання окремого php-файлу для кожної сторінки додатку.

Але в об’єктно-орієнтованому програмуванні використовується активна модель MVC, де модель – це не тільки сукупність коду доступу до даних і СУБД, але і вся бізнес логіка. Варто зазначити про можливості моделі інкапсулювати в собі інші моделі. В свою чергу контролери являють собою лише елементи системи, в безпосередні обов’язки яких входить прийом даних із запиту і передача їх іншим елементам системи. Тільки у цьому випадку контролер стає «тонким» і виконує виключно функції зв’язкової ланки(glue layer) між певними компонентами системи.

## Реалізація

Концепція MVC вперше застосувалася при проектуванні мови програмування Smalltalk як модель для інтерфейсу користувача . Зараз ця концепція використовується у багатьох об’єктно-орієнтованих мовах програмування, пристворенні складних систем.

### Java

У мові програмування Java концепція MVC підтримується на рівні стандартних класів-бібліотек. В результаті використання парадигми MVC програміст отримує в своє розпорядження могутню структуру об'єктів-компонентів, функції яких чітко розмежовані, що гарантує надійність і розширюваність системи, що розробляється.

• Десктоп-каркаси: Swing, JFace

# Проектування програмного забезпечення

## Use case

### Реєстрація нового користувача

Студент реєструється самостійно, заповнюючи реєстраційну форму, вводить усі необхідні дані (логін, пароль, підтвердження паролю, ім’я, прізвище, по-батькові, e-mail, дійсний контактний телефон, контакти батьків, адресу проживання, ін. ). Реєстрація завершується підтвердженням адміністратора.

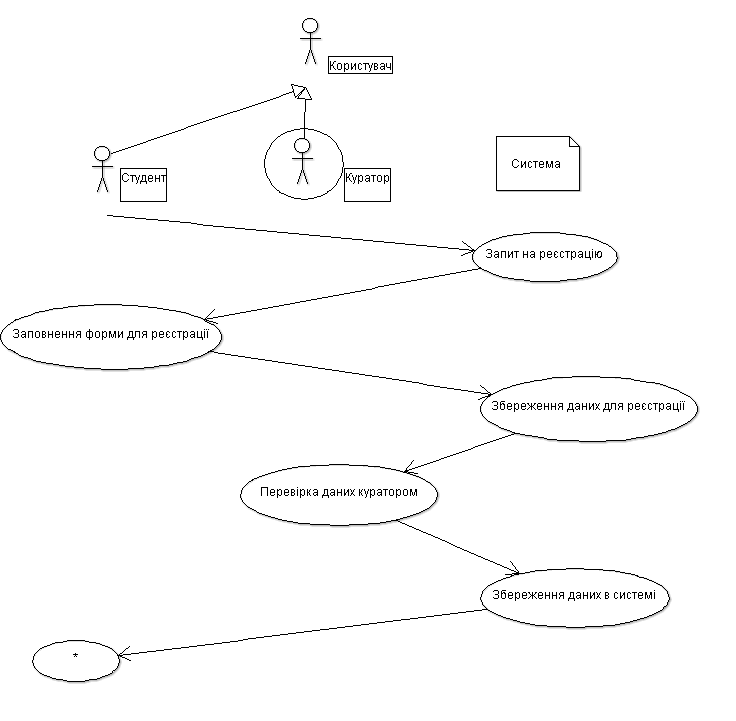
Допрацювати діаграму: куратор не пов’язаний із системою

Рисунок 1.2 Реєстрація нового користувача

1. У початковому вікні програми натиснути кнопку «Реєстрація», у вікні зявиться реєстраційна форма
2. Ввести персональні дані у реєстраційну форму, натиснути кнопку «Зареєструвати»
3. Реєстрація завершується перевіркою введених даних адміністратором

### Управління профілем

**Передумови** : студента ідентифіковано та авторизовано

**Сценарій**:

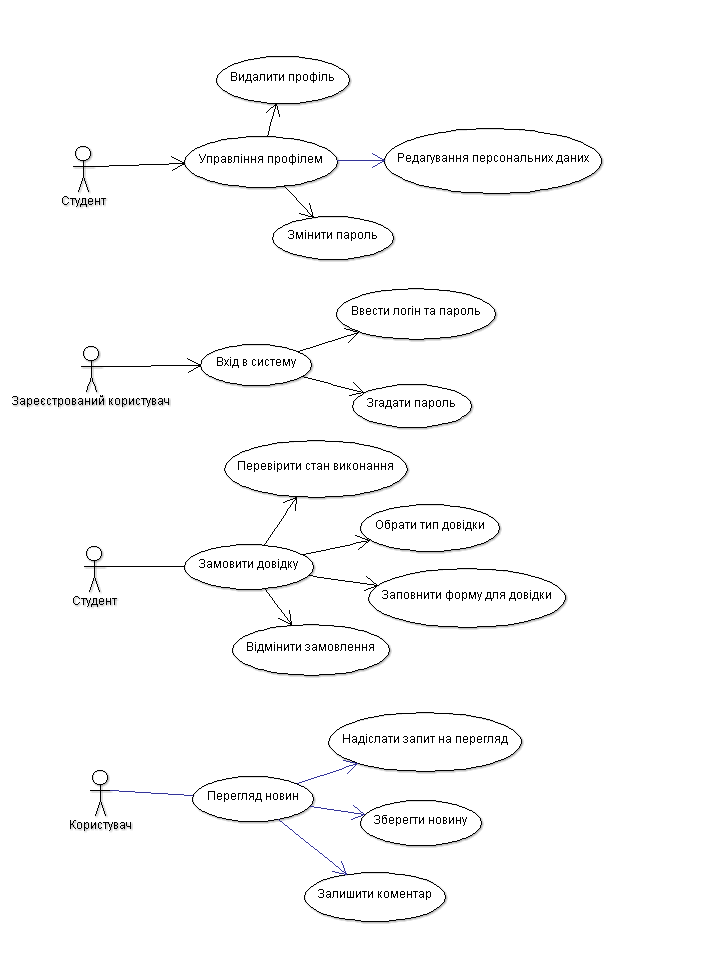
1. У меню програми натисннути кнопку «Управління профілем»
2. Обрати пункт «Редагувати акаунт» або «Видалити акаунт»
3. Якщо обрано пункт «Редагувати акаунт»:
4. Користувачу надаєтья форма для зміни персональних даних та паролю
5. Користувач вносить зміни і натискає кнопку «Підтвердити»
6. Якщо обрано пункт «Видалити акаунт»:
7. Висвічується діалогове вікно для підтвердження намірів видалення профілю
8. Користувач підтверджує або скасувує видалення

Рисунок 1.3 Управління профілем

### Замовлення довідки

**Передумови** : студента ідентифіковано та авторизовано

1. У меню програми натисннути кнопку «Замовити довідку»
2. Обрати тип довідки із запропонованих
3. Ввести потрібні дані
4. Натиснути кнопку «Замовити»

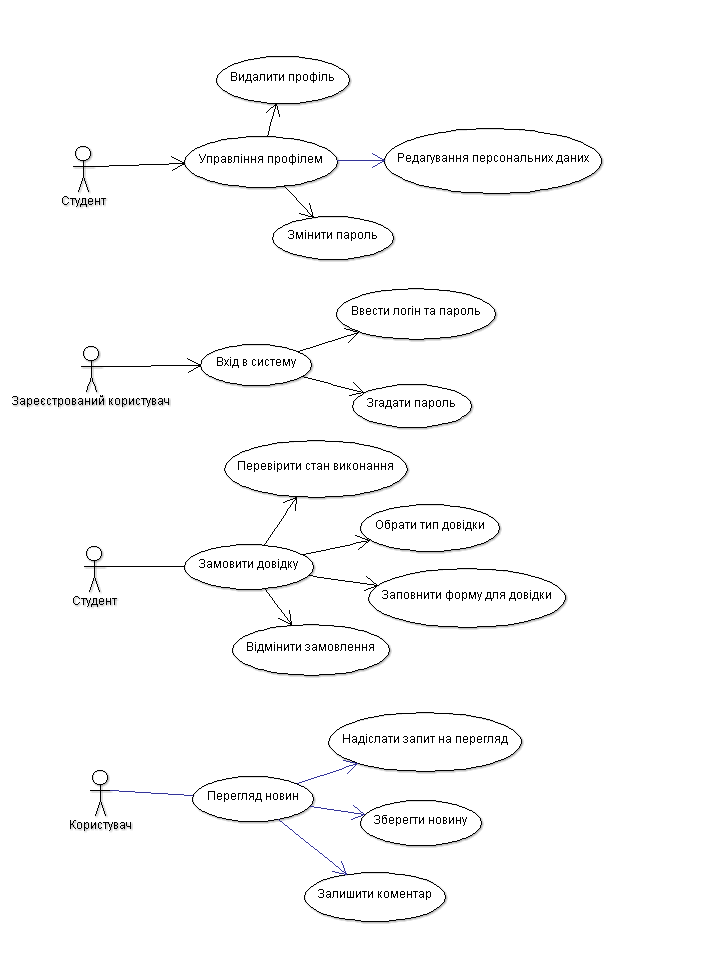


Рисунок 1.4 Замовлення довідки

### Залишити повідомлення для викладача/ групи

**Передумови** : студента ідентифіковано та авторизовано

1. У меню програми натисннути кнопку «Залишити повідомлення»
2. Заповнити форму для повідомлення
3. Відправити

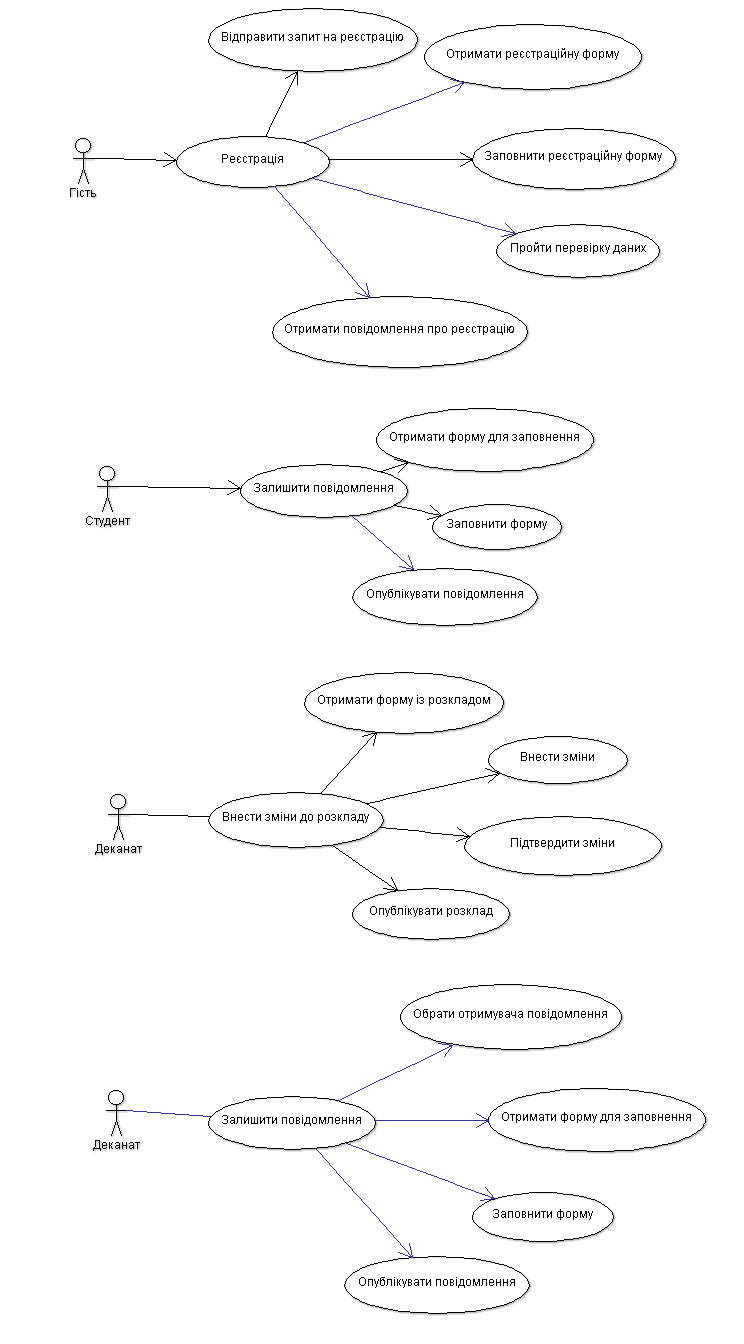


Рисунок 1.5 Залишення повідомлення

### Пергляд розкладу студента/викладача

**Передумови** : студента ідентифіковано та авторизовано

1. У меню програми натисннути кнопку «Розклад»
2. Обрати тип розладу(розклад следентів, розклад викладачів)
3. Ввести назву навчальної групи/прізвище і’мя викладача
4. Натиснути «Пошук»
5. Отримати розклад

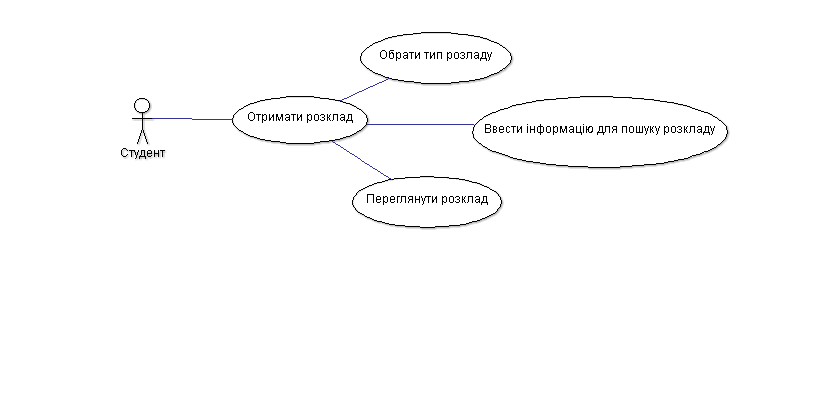


Рисунок 1.6 Отримання розкладу

### Авторизація у системі

**Передумови** : користувач зареєстрований

1. Відкрити вікно програми
2. Ввести логін і пароль у відповідні поля
3. Натиснути кнопку «Увійти»

Рисунок 1.7 Авторизація

## Проектування графічного інтерфейсу користувача

### Макет графічного інтерфейсу

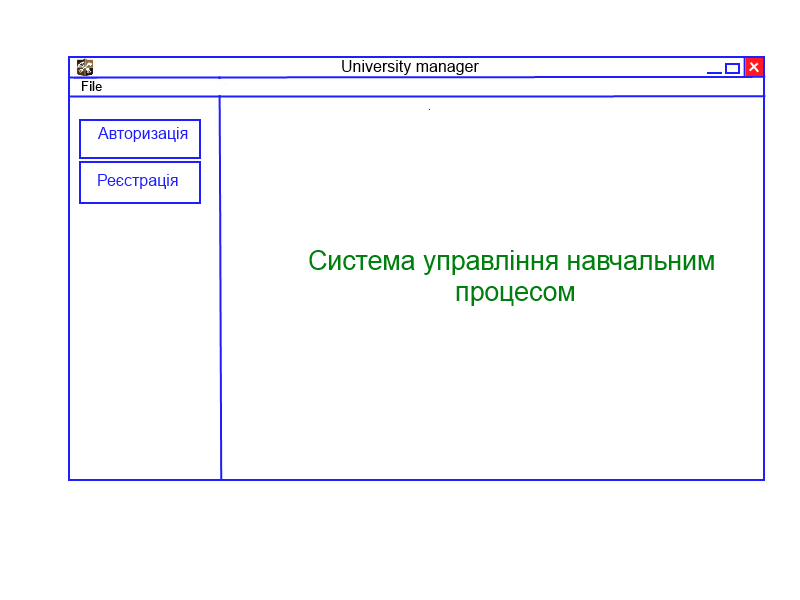
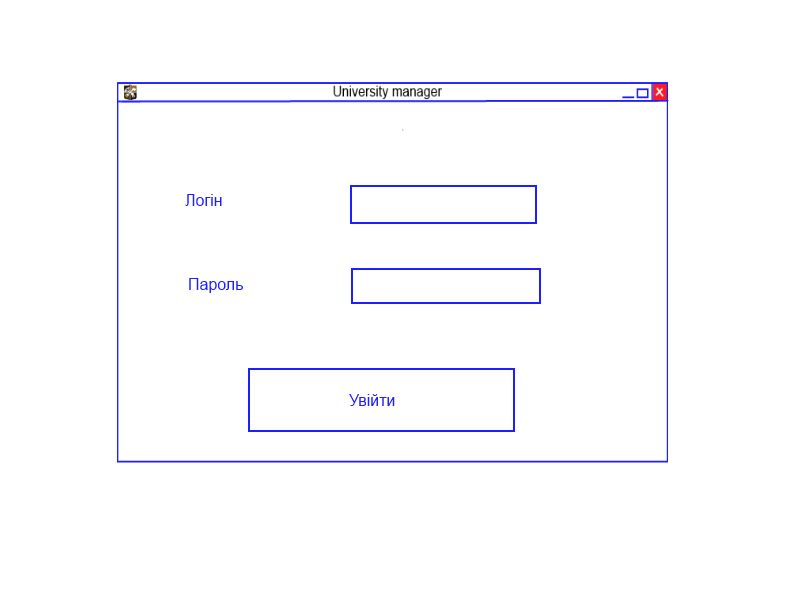
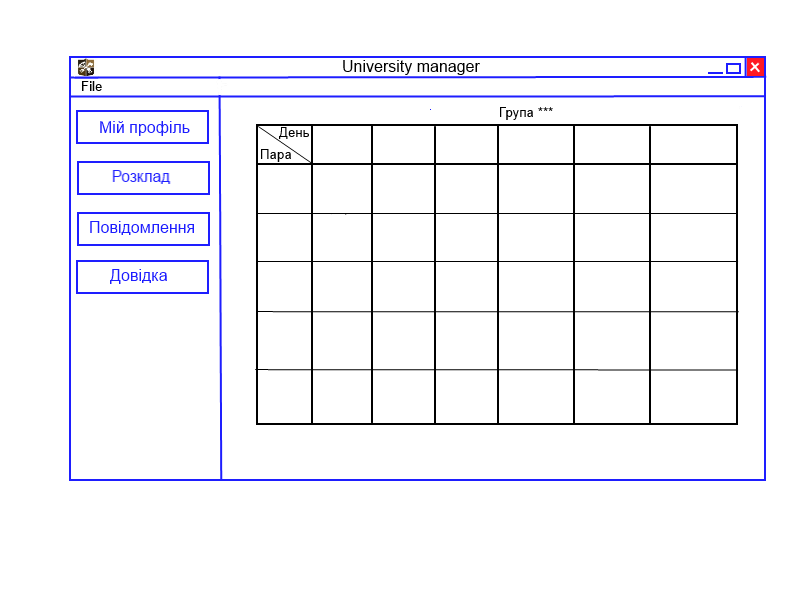


Рисунок 1.7 Макет посаткового вікна програми

  
Рисунок 1.8 Макет вікна авторизації

Рисунок 1.9 Макет вікна отримання розкладу

## Діаграма граничних класів

Рисунок 1.10 Діаграма граничних класів