НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

ННК "Інститут прикладного системного аналізу"
(повна назва інституту/факультету)

Кафедра Системного проектування
(повна назва кафедри)

«До захисту допущено»

Завідувач кафедри
_____ А.І.Петренко
(підпис) (ініціали, прізвище)

"___"_____2016 р.

Дипломна робота
першого (бакалаврського) **рівня вищої освіти**
(першого (бакалаврського), другого (магістерського))

зі спеціальності 7.050102, 8.050102 Інформаційні технології проектування
7.050103, 8.050103 Системне проектування
(код та назва спеціальності)

на тему: Інтелектуальна система створення індивідуального календарного плану навчального процесу

Виконав: студент IV курсу, групи ДА-22
(шифр групи)

Акимов Вадим Сергійович _____
(прізвище, ім'я, по батькові) (підпис)

Керівник к.т.н., доцент Кисельов Г.Д. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант з економіки _____ _____ _____
(назва розділу) (посада, вчене звання, науковий ступінь, прізвище, ініціали) (підпис)

Рецензент _____ _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Нормоконтроль ст. викладач Бритов О.А. _____
(підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2016 року

Національний технічний університет України
«Київський політехнічний інститут»

Факультет (інститут) ННК "Інститут прикладного системного аналізу"
(повна назва)

Кафедра Системного проектування
(повна назва)

Рівень вищої освіти Перший(Бакалаврський)
(перший (бакалаврський), другий (магістерський) або спеціаліста)

Спеціальність 7.05010102, 8.05010102 Інформаційні технології проектування
7.05010103, 8.05010103 Системне проектування
(код і назва)

ЗАТВЕРДЖУЮ

Завідувач кафедри
_____ А.І.Петренко
(підпис) (ініціали, прізвище)

«___»_____2016 р.

ЗАВДАННЯ

на дипломний проект (роботу) студенту

Акимову Вадиму Сергійовичу

(прізвище, ім'я, по батькові)

1. Тема проекту (роботи)) Інтелектуальна система створення індивідуального календарного плану навчального процесу

керівник проекту (роботи)) Кисельов Геннадій Дмитрович, к.т.н., доцент,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від 12 травня 2016 р. № 50-ст

2. Строк подання студентом проекту (роботи) 15.06.2016

3. **Вихідні дані до проекту** (роботи):

- Теорія розкладів як розділ дослідження операцій.
- Стохастичні та евристичні алгоритми для вирішення задачі глобальної оптимізації календарного плану.
- Алгоритм імітації відпалу.
- Жадібний алгоритм.
- Технологія Spring MVC для розробки веб-орієнтованого прототипу інтелектуальної системи створення індивідуального календарного плану навчального процесу.

4. **Перелік питань, які повинні бути розроблені**

1. Розгляд класифікації задач, що вирішуються в рамках одного з розділів дослідження операцій - теорії розкладів.

2. Аналіз існуючих рішень для формування та оптимізації календарних планів (навчальних розкладів).

3. Опис предметної області. Розробка вимог до характеристик прототипу системи створення календарного плану навчального процесу.

4. Опис математичної моделі та особливостей реалізації алгоритмів (алгоритму імітації відпалу, жадібного алгоритму).

5. Опис обраних технологій для реалізації прототипу.

6. Вибір технологій та реалізація прототипу інтелектуальної системи створення індивідуального календарного плану навчального процесу.

7. Аналіз ефективності алгоритмів. Створення порівняльної характеристики отриманих результатів та основних показників роботи обох алгоритмів (алгоритму імітації відпалу та жадібного алгоритму).

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслеників, плакатів тощо)

1.UML діаграма класів – плакат.

2. Веб інтерфейс прототипу системи – плакат.

3. Результати роботи програми – плакат.

6. Консультанти розділів проекту (роботи)*

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Економічна частина	проф. док. ек. н. Семенченко Н. В.		

7. Дата видачі завдання 01.02.2016.

* Консультантом не може бути зазначено керівника дипломного проекту (роботи).

Календарний план

№ з/п	Назва етапів роботи та питань, які повинні бути розроблені відповідно до завдання	Термін виконання	Позначки керівника про виконання завдань
1	Планування дипломного проекту	14.04.2016 18.04.2016	
2	Опис предметної області	18.04.2016 23.04.2016	
3	Дослідження існуючих рішень	23.04.2016 26.04.2016	
4	Розробка технічного завдання	26.04.2016 01.05.2016	
5	Реалізація прототипу системи календарного планування	01.05.2016 22.05.2016	
6	Опис стохастичних та евристичних алгоритмів	01.05.2016 02.05.2016	
7	Реалізація обраних алгоритмів	02.05.2016 11.05.2016	
8	Оформлення роботи	24.05.2016 29.05.2016	
9	Підготовка презентації дипломного проекту	29.05.2016 11.06.2016	

Студент _____ В.С.Акимов
(підпис) (ініціали, прізвище)

Керівник проекту (роботи) _____ Г.Д.Кисельов
(підпис) (ініціали, прізвище)

АНОТАЦІЯ

бакалаврської дипломної роботи Акимова Вадима Сергійовича
на тему «Інтелектуальна система створення індивідуального календарного плану навчального процесу»

Об’єктом дослідження в даній роботі є прототип інтелектуальної системи, яка дозволяє формувати календарний план навчального процесу з урахуванням індивідуального графіку занятості студента. Основною метою даної роботи було дослідження оптимальності використання стохастичного алгоритму імітації відпалу та евристичного жадібного алгоритму для вирішення задачі багатокритеріальної оптимізації календарного плану навчального процесу.

На сьогоднішній день проблема формування календарного плану часто виникає у студентів, які намагаються поєднувати навчання в університеті, роботу, власних духовний та фізичний розвиток, відпочинок. Виникає необхідність у сервісі, який виконує планування календарного графіку виконання робіт з врахуванням власних справ студента, підвищуючи його ефективність. Головна мета такого сервісу – забезпечити своєчасну здачу запланованих навчальним планом робіт. Усі наявні рішення не враховують індивідуальний розклад студента та рівень його щоденної завантаженості власними справами при формуванні графіку виконання навчального плану.

Результат роботи – це прототип системи створення та оптимізації індивідуального календарного плану виконання навчального процесу, порівняльна характеристика основних показників роботи обох алгоритмів та демонстрація результатів оптимізації календарного плану виконання навчального процесу на 1 семестр.

Загальний обсяг роботи 82 с., 35 рис., 6 табл., 27 посилань.

Ключові слова: оптимізація індивідуального календарного плану, теорія розкладів, стохастичні алгоритми, евристичні алгоритми, алгоритм імітації відпалу, функція енергії, температура, жадібний алгоритм.

АННОТАЦИЯ

бакалаврской дипломной работы Акимова Вадима Сергеевича
на тему: «Интеллектуальная система создания индивидуального календарного плана учебного процесса»

Объектом исследования в данной работе является прототип интеллектуальной системы, которая позволяет формировать календарный план учебного процесса с учетом индивидуального графика занятости студента. Основной целью данной работы было исследование оптимальности использования стохастического алгоритма имитации отжига и эвристического жадного алгоритма для решения задачи многокритериальной оптимизации календарного плана учебного процесса.

На сегодняшний день проблема формирования календарного плана часто возникает у студентов, которые совмещают учебу в университете, работу, собственных духовное и физическое развитие, отдых. Возникает необходимость в сервисе, который выполняет планирование календарного графика выполнения работ с учетом собственных дел студента, повышая его эффективность. Главная цель такого сервиса - обеспечить своевременную сдачу запланированных учебным планом работ. Все имеющиеся решения не учитывают индивидуальное расписание студента и уровень его ежедневной загруженности собственными делами при формировании графика выполнения учебного плана.

Результат работы - это прототип системы создания и оптимизации индивидуального календарного плана выполнения учебного процесса, сравнительная характеристика основных показателей работы обеих алгоритмов и демонстрация результатов оптимизации календарного плана выполнения учебного процесса на 1 семестр.

Общий объем работы: 82 с., 35 рис., 6 табл., 27 источников.

Ключевые слова: оптимизация индивидуального календарного плана, теория расписаний, стохастические алгоритмы, эвристические алгоритмы, алгоритм имитации отжига функция энергии, температура, жадный алгоритм.

ANNOTATION

on Vadym Akymov bachelor’s degree
thesis: “Intellectual system for creating an individual schedule of educational process”

The object of study in this work is a prototype of intelligent system that allows you to create the schedule of the educational process taking into account individual student affairs. The main purpose of this work was to study the optimal usage of stochastic annealing simulation algorithm and heuristic greedy algorithm for solving multi-criterial optimization schedule of the educational process.

Today many students are often faced with the schedule planning problem trying to combine university studies, work, their spiritual and physical development and recreation. There is a need for a service that performs planning of educational works with regard to the student affairs increasing theirs efficiency. The main purpose of such service is to ensure that planned curriculum has been completed in time and all efforts of each student were distributed effectively. All existing solutions ignore individual student's daily workload level while forming the schedule.

The result of this work is a prototype of the system that allows creating and optimizing individual schedule of the educational process, comparative characteristics of the main parameters of both algorithms and demonstration of the optimization results of the educational process for one semester.

Bachelor's work size: 82 p.., 35 pic.., 6 tables., 27 sources.

Keywords: optimization of individual schedule, scheduling theory, stochastic algorithms, heuristic algorithms, simulated annealing algorithm, the function of energy, temperature, greedy algorithm.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, ТЕРМІНІВ

ПЗ – програмне забезпечення.

API – прикладний програмний інтерфейс (англ. application programming interface).

REST – Передача стану подання (англ. Representation State Transfer).

XML – Розширювана мова розмітки (англ. Extensible Markup Language).

UI – засіб зручної взаємодії користувача з інформаційною системою (англ. User Interface).

SQL (мова структурованих запитів) — декларативна мова програмування для взаємодії користувача з базами даних, що застосовується для формування запитів, оновлення і керування реляційними БД, створення схеми бази даних і її модифікації, системи контролю за доступом до бази даних (англ. Structured query language).

MVC — архітектурний шаблон, який використовується під час проектування та розробки програмного забезпечення (англ. Model-view-controller).

БД – база даних (англ. database).

IoC — це принцип побудови програми, при якому її частини отримують потік керування (викликаються) із загальної спільно використовуваної бібліотеки (англ. Inversion of Control, IoC).

ВСТУП

Люди протягом усього життя вирішують задачу планування власного часу. Кожна задача потребує різних часових інвестицій та повинна бути виконана до певної дати. Виходячи з цих даних, усі справи розміщуються послідовно у часі, формуючи розклад. Усі керуються при цьому “схожими алгоритмами”, намагаючись виконати всі важливі справи вчасно, виділивши при цьому максимальний час на дозвілля та відпочинок.

Часто проблема формування календарного плану виникає у студентів, які намагаються поєднувати навчання в університеті, роботу, власний духовний та фізичний розвиток та відпочинок. Виникає необхідність у сервісі, який виконує планування календарного графіку виконання робіт з врахуванням власних справ студента, підвищуючи його ефективність та переймаючи цей важкий тягар з його плечей. Головна мета такого сервісу – забезпечити своєчасну здачу запланованих навчальним планом робіт, ефективно розподіляючи найважливіший ресурс кожної людини – час.

Часто ми плануємо наші дії в порядку зростання крайніх термінів виконання робіт (справ). Наприклад, студенти під час екзаменаційної сесії вчать предмет з найменшим директивним терміном (найближчий за датою здачі іспит). Якщо перший іспит необхідно здати 12-го січня, другий 15-го, а третій 19-го, при цьому на кожен іспит необхідно 3 дні, то більшість студентів складуть такий розклад: "з 9 по 11 підготовка до першого іспиту, з 12 по 14 до другого і з 16 по 18 до третього ".

Складнощі при формуванні розкладів з'являються тоді, коли робіт стає багато, потрібно врахувати безліч додаткових умов і / або скласти розклад не для однієї людини, а для цілого колективу, коли існує сотні робіт і виконавців.

У процесі вирішення таких завдань, були вироблені спільні рекомендації, принципи і методики складання розкладів. Надалі подібні завдання стали досліджуватися в рамках спеціального розділу науки - теорії розкладів.

Ця наука виникла не на порожньому місці, а виникла на стику інших областей наукового знання. Перш ніж дати визначення, що таке теорія розкладів, опишемо одну важливу область математики.

Дослідження операцій (ДО) - науковий метод вироблення кількісно обґрунтованих рекомендацій щодо прийняття рішень. Важливість кількісного фактора в ДО і цілеспрямованість сформульованих рекомендацій дозволяють визначити ДО як теорію прийняття оптимальних рішень. ДО сприяє перетворенню мистецтва прийняття рішень в математичну дисципліну. Термін "ДО" виник в результаті буквального перекладу виразу "operation research", введеного в кінці 30-х років 20-го століття як умовне найменування одного з підрозділів британських ВПС, що займається питаннями використання радіолокаційних установок в загальній системі оборони. Спочатку ДО було пов'язано з вирішенням завдань військового змісту, але вже з кінця 40-х років минулого століття воно використовується для вирішення технічних, техніко-економічних завдань, а також завдань управління на різних рівнях.

Тепер ми можемо дати визначення теорії розкладів.

Теорія розкладів - це розділ дослідження операцій, в якому будуються і аналізуються математичні моделі календарного планування (тобто упорядкування в часі) різних цілеспрямованих дій з урахуванням цільової функції і різних обмежень.

Змістовно багато завдань ТР є оптимізаційними, тобто складаються у виборі (знаходженні) серед безлічі допустимих розкладів (розкладів, що допускаються умовами завдання) тих рішень, на яких досягається "оптимальне" значення цільової функції. Зазвичай під "оптимальністю" розуміється мінімальне або максимальне значення деякої цільової функції. Допустимість розкладу розуміється в сенсі його здійсненності, а оптимальність - в сенсі його доцільності.

Приклад. Необхідно побудувати будинок якомога швидше, при цьому послідовність робіт повинна бути дотримана. Для даної задачі допустимий розклад то, при якому буде дотримана послідовність робіт, а оптимальний

розклад - це допустимий розклад, при якому будинок буде побудований в мінімальні терміни.

Календарний план **–** проектно-технологічний документ, який визначає послідовність, інтенсивність, тривалість робіт, також їх взаємозв'язок з іншими видами робіт, необхідність в матеріальних, технічних, трудових або фінансових ресурсах.

Основна задача планування – полягає в створенні певного порядку дій та оптимальному розподілі ресурсів (вільний час студента), що необхідні для досягнення поставленої мети (успішне та своєчасне виконання студентом усіх робіт, що передбачені навчальним процесом з врахуванням власного графіку). Сформований графік робіт повинен задовольняти всім наявним hard stop критеріям, а також якомога менше порушувати умови soft stop критеріїв.

ПОСТАНОВКА ЗАДАЧ

Проблема планування виконання великої кількості навчальних задач з врахуванням різних обмежуючих критеріїв спричиняє необхідність в інтелектуальних системах створення та оптимізації календарних планів. На сьогоднішній день на ринку представлено безліч рішень-аналогів. Багато з них мають такі суттєві переваги як створення оптимізованого розкладу, що відповідає усім заданим критеріям, імпорт даних з різних форматів, підтримка мобільних клієнтів, ручні налаштування розкладу. Проте жодна з існуючих систем не орієнтована на врахування індивідуального розкладу користувачів при формуванні навчального плану. Тому виникає необхідність розробки такої системи, яка б забезпечила інтеграцію з найпопулярнішими рішеннями в області планування власних справ. В даній роботі розглядається інтеграція з одним із таких сервісів – Google Calendar.

Наступною задачею є вивчення ефективності застосування стохастичних та евристичних алгоритмів оптимізації календарного плану навчального процесу. Для вирішення даної проблеми були обрані такі алгоритми: алгоритм імітації відпалу та жадібний алгоритм. Результатом аналізу є порівняльна характеристика отриманих рішень у вигляді графіків, що відображають основні показники роботи алгоритмів.

Важливим кроком є опис предметної області та розробка основних вимог до характеристик розроблюваного прототипу системи створення індивідуального календарного плану навчального процесу. Основним інструментом опису предметної області є UML діаграми прецедентів, діаграми послідовностей та DataFlow діаграми.

Результатом даної роботи має бути розроблений прототип системи, що включає в себе реалізацію обраних алгоритмів та модульні тести, що забезпечують контроль якості отриманого рішення. Демонстрація результатів роботи готового програмного забезпечення в даній роботі проводиться на прикладі реальних навчальних задач одного навчального семестру.

1. АНАЛІТИЧНИЙ ОГЛЯД

Теорія розкладу являє собою один із розділів дослідження операцій. Даний напрямок бере свій початок з відомої роботи Генрі Ганта 1903р, який запропонував те, що сьогодні називають діаграмою Ганта. Методи і алгоритми вирішення задач теорії розкладу використовуються для вирішення задач комбінаторної оптимізації.

1.1 Класифікація завдань теорії розкладів

Наведемо деякі способи класифікації задач ТР:

- За типом шуканого рішення:
  - Завдання упорядкування. У цих завданнях вже задано розподіл робіт за виконавцями, а також визначені всі параметри робіт (тривалість виконання, час надходження та т.д.). Необхідно скласти розклад (або порядок) виконання робіт кожним виконавцем [2].
  - Завдання узгодження. Основна увага в цих завданнях приділяється вибору тривалості виконання робіт, часу надходження і іншим параметрам;
  - Завдання оптимального розподілу робіт за виконавцями.
- За типом цільової функції:
  - Завдання з сумарними критеріями оптимізації. Наприклад мінімізація сумарного значення моментів закінчення обслуговування робіт $\sum_{j=1}^{n} C_j$ (1.1).
  - Завдання з min/max (мінімаксними) критеріями оптимізації. Відмінність цих завдань від завдань з сумарними критеріями полягає в тому, що потрібно мінімізувати не суму деяких значень, а лише максимальне з них. Наприклад, якщо в згаданій задачі необхідно мінімізувати максимальне значення Cmax, де Cmax = max(j∈N) Cj, то ми отримаємо одну з тривіальних завдань цього класу.

- Багатокритеріальні задачі оптимізації. Якщо в досліджуваних завданнях необхідно побудувати оптимальне рішення з точки зору декількох цільових установок (функцій), то такі завдання називаються багатокритеріальними. Наприклад, якщо в згаданій задачі необхідно не тільки мінімізувати значення (1.1), але й мінімізувати і час простою процесора (приладу), то це багатокритеріальна задача.
- Завдання на побудову допустимого розкладу. В попередньому розділі було дано приклад такого завдання. Необхідно відзначити, що даний клас задач можна звести до оптимізаційних задач, ввівши спеціальну функцію штрафу, яку потрібно мінімізувати. Проте, прийнято виділяти такі задачі в окремий клас.

- За способом завдання вхідної інформації:
  - Детерміновані задачі (off-line). Для таких задач характерно, що всі вхідні дані завдання точно відомі, тобто дані значення всіх параметрів до початку її вирішення.
  - Динамічні задачі (on-line). Для даних завдань розклад будується в режимі реального часу, тобто перед початком виконання завдання значення всіх параметрів невідомі. Розклад будується частинами в міру надходження нової інформації. При цьому в будь-який момент може бути знадобитися відповідь про якість побудованого "часткового" розкладу.
- Згідно розділу ТР. В рамках ТР прийнято виділяти наступні розділи:
  - Планування мереж або побудова розкладу для проекту, Project scheduling (PS).
  - Календарне планування або побудова розкладу для приборів, Machine scheduling (MS).
  - Складання тимчасових таблиць (Time Tabling).
  - Доставка товарів в магазини (Shop-Floor Scheduling).

- Складання розкладів руху транспортних засобів (Transport Scheduling), Циклічні розкладу для транспортних засобів (Vehicle Routing).
- Складання розкладів спортивних заходів (Sports scheduling).

1.2 Аналіз існуючих рішень

В даному розділі розглянемо три існуючі рішення (Automated College Timetable Generator, aSc Timetables, Timetableweb) в області формування та оптимізації календарних планів., наведемо їх переваги та недоліки та на основі отриманої інформації сформуємо вимоги до власного прототипу.

1.2.1 Automated College Timetable Generator (Nevon Projects Software)

Проект пропонує автоматичну генерацію календарного розкладу для навчальних закладів з використанням генетичного алгоритму. Основною перевагою проекту є те, що він гарантує формування розкладу без перекриття часових інтервалів виділених для учбових класів.

Опис додатку та проблем що він вирішує.

Більшість коледжів мають ряд різних курсів і кожен курс має цілий ряд предметів. За основу береться, що існує обмежена кількість факультетів, кожен з них викладає більше одного предмету. Так що тепер задачею є знайти розклад при заданому часовому інтервалів таким чином, щоб хронометраж різних предметів не перекривав один одного і графік роботи був оптимально спланований на протязі всього навчального періоду.

Для цього використовується генетичний алгоритм. В алгоритмі ми виділяємо покоління розкладів, тобто головною задачею є вибір найкращого розкладу у ході еволюції. Цей об'єкт складається з об'єктів класів (аудиторій). Фітнес-оцінка зв’язана з відсотком колізій часових інтервалі кожного класу.

Об'єкт клас складається з об'єктів тижні. Об’єкт тиждень складається з днів. В свою чергу дні складаються з часових інтервалів. Часовий інтервал зберігає в собі предмет, викладача та групу студентів, які відвідують даний предмет.

Додаток має інтерфейс з великою кількістю конфігурації, що дозволяє гнучко налаштувати всі параметри розкладу користувачем.

Інтерфейс користувача

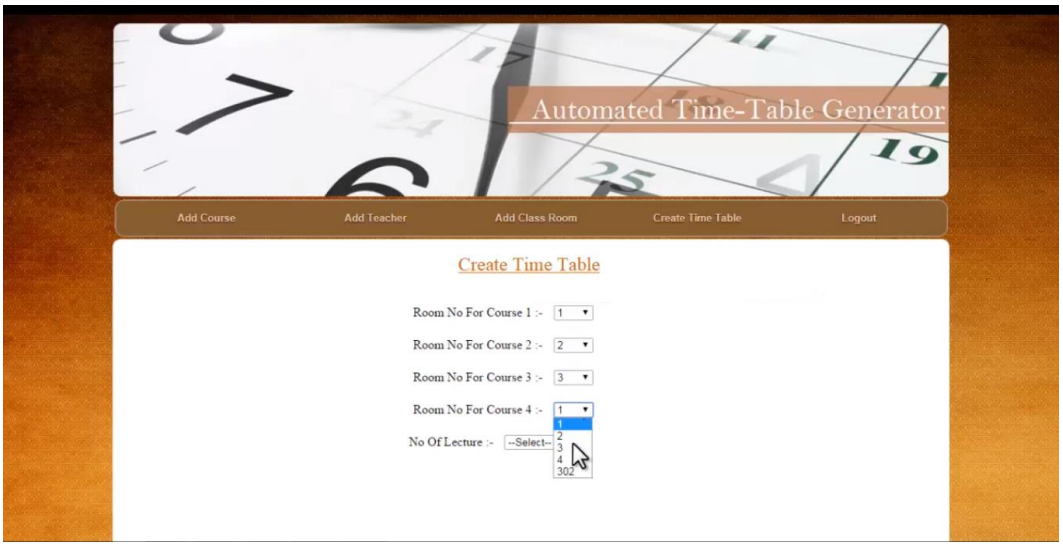


Рис.1.1 Вхід до сервісу

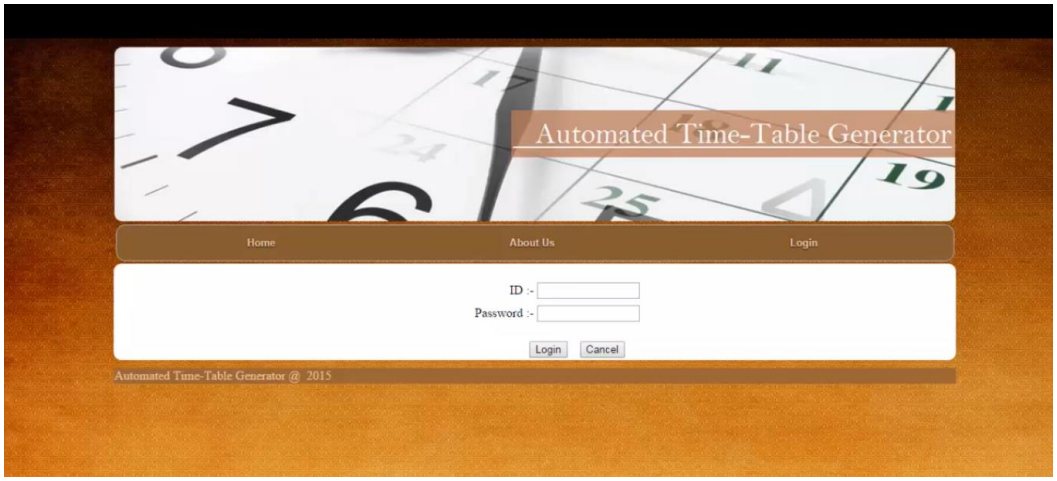
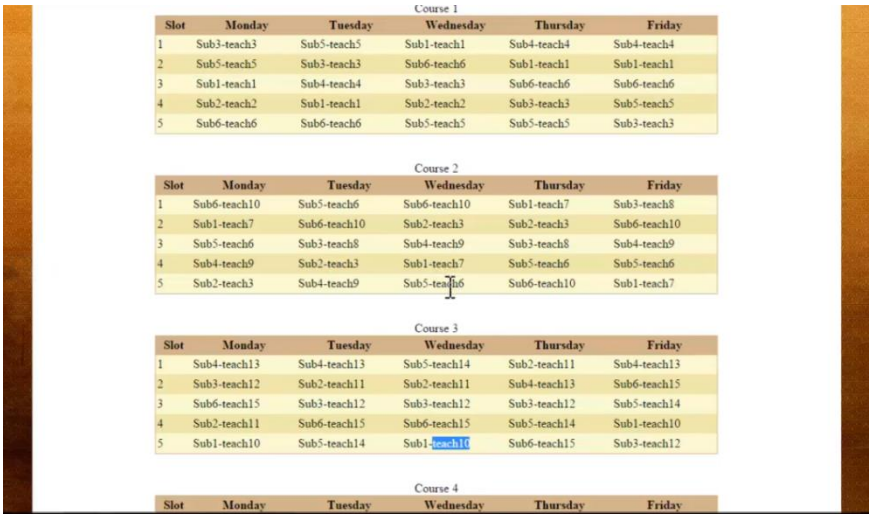


Рис.1.2 Інтерфейс вводу



Course 1

Slot	Monday	Tuesday	Wednesday	Thursday	Friday
1	Sub3-teach3	Sub5-teach5	Sub1-teach1	Sub4-teach4	Sub4-teach4
2	Sub5-teach5	Sub3-teach3	Sub6-teach6	Sub1-teach1	Sub1-teach1
3	Sub1-teach1	Sub4-teach4	Sub3-teach3	Sub6-teach6	Sub6-teach6
4	Sub2-teach2	Sub1-teach1	Sub2-teach2	Sub3-teach3	Sub5-teach5
5	Sub6-teach6	Sub6-teach6	Sub5-teach5	Sub5-teach5	Sub3-teach3

Course 2

Slot	Monday	Tuesday	Wednesday	Thursday	Friday
1	Sub6-teach10	Sub5-teach6	Sub6-teach10	Sub1-teach7	Sub3-teach8
2	Sub1-teach7	Sub6-teach10	Sub2-teach3	Sub2-teach3	Sub6-teach10
3	Sub5-teach6	Sub3-teach8	Sub4-teach9	Sub3-teach8	Sub4-teach9
4	Sub4-teach9	Sub2-teach3	Sub1-teach7	Sub5-teach6	Sub5-teach6
5	Sub2-teach3	Sub4-teach9	Sub5-teach6	Sub6-teach10	Sub1-teach7

Course 3

Slot	Monday	Tuesday	Wednesday	Thursday	Friday
1	Sub4-teach13	Sub4-teach13	Sub5-teach14	Sub2-teach11	Sub4-teach13
2	Sub3-teach12	Sub2-teach11	Sub2-teach11	Sub4-teach13	Sub6-teach15
3	Sub6-teach15	Sub3-teach12	Sub3-teach12	Sub3-teach12	Sub5-teach14
4	Sub2-teach11	Sub6-teach15	Sub6-teach15	Sub5-teach14	Sub1-teach10
5	Sub1-teach10	Sub5-teach14	Sub1-teach10	Sub6-teach15	Sub3-teach12

Course 4

Slot	Monday	Tuesday	Wednesday	Thursday	Friday

Рис.1.3 Результати планування

Переваги

- Факультетам не потрібно турбуватися про часові конфлікти у розкладі
- Реалізована у формі веб сервісу, тому користувачу нічого не потрібно встановлювати щоб скористуватись ним

Недоліки

- Користувач повинен від форматувати розклад трохи після того, як він буде створений.
- Дуже вузько направлений сервіс планування
- Велика ціна (170$) для такого простого сервісу
- Відсутність будь якої документації та вагомих аргументів щодо переваг даного продукту
- Погано оформлений інтерфейс

1.2.2 aSc Timetables

Створення розкладу для школи може бути досить складною задачею, особливо маємо справу з великою кількістю класів, підкласів, вчителів.

aSc Timetables це програма, яка пропонує вам можливість легко і швидко створювати графіки з класами і викладачами. Додаток покриває всі можливі види організації навчального процесу, наприклад таких як додавання різних вчителів до того ж класу. Програма може працювати враховуючи індивідуальну конфігурацію для класів та вчителів, попереджаючи вас, якщо викладач може призначити свій предмет у точно визначений інтервал часу, або якщо деякий вчитель доступний у вівторок вранці, наприклад. Після того, як ви ввели інформацію, програма генерує повний розклад протягом декількох хвилин. Це буде перелік класів, які заповнюють весь тиждень з даними про аудиторії, вчителів та інша необхідна інформація

Додаток також може перевірити накладені обмеження у розкладі і допоможе вам уникнути стандартних помилок, дозволяючи вносити зміни вручну, попереджаючи, якщо ви допустили помилку.

Інтерфейс користувача

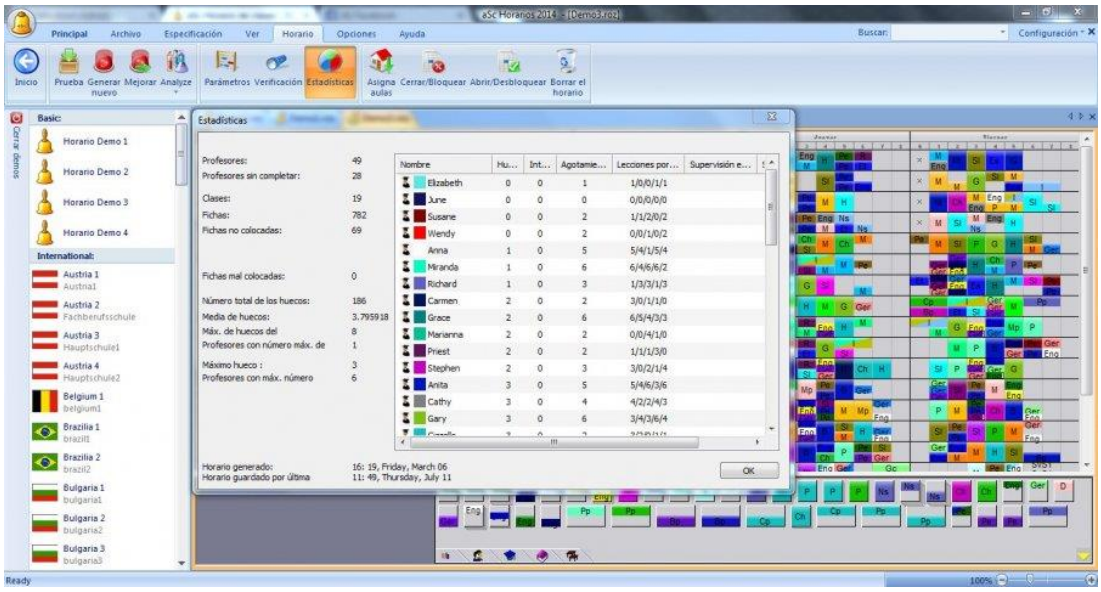


Рис.1.4 Інтерфейс створення нового розкладу

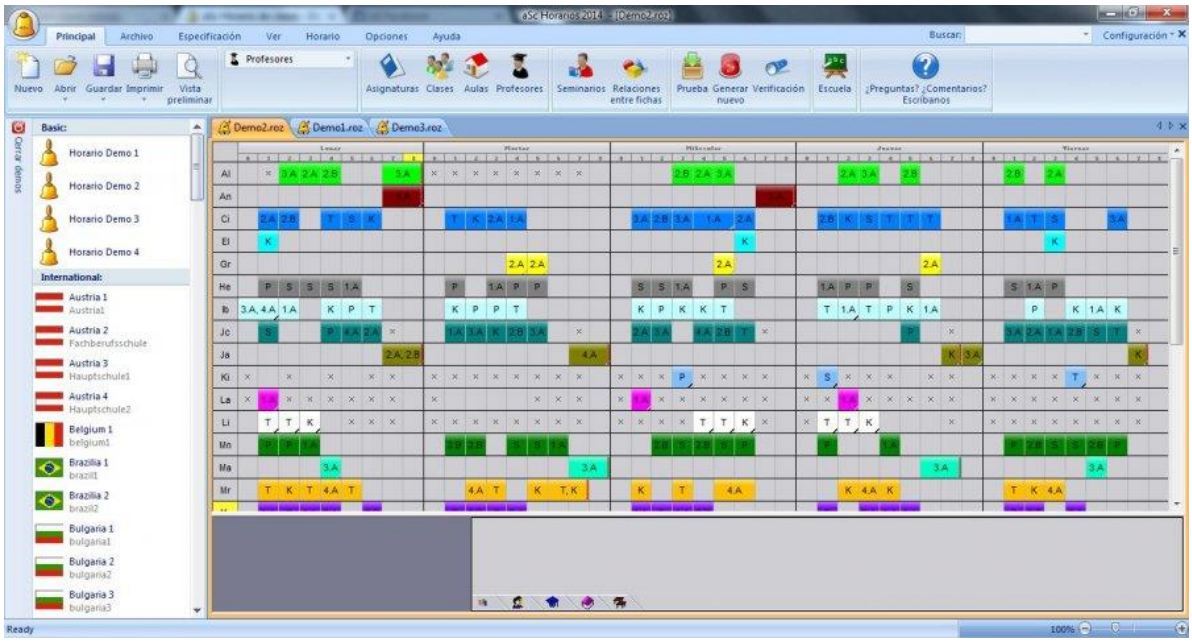


Рис. 1.5 Інтерфейс перегляду розкладу

Переваги

- Ручні налаштування. Створений програмою розклад можна змінювати і вручну
- Перевірка розкладу. Алгоритм програми перевірить розклад і повідомить про будь-які невідповідності, не виконані умови
- Просте введення даних. Введення вихідних даних дуже просте і швидке

- Розклад в мобільних телефонах. Можете публікувати розклад в Інтернеті і зробити доступним для смартфонів або планшетних комп'ютерів
- Імпорт даних. У Вас, напевно, вже є більшість необхідних даних для розкладу в електронному форматі (списки вчителів, кабінетів, класів, предметів) - їх можна просто імпортувати в програму
- Шкільні будівлі. У програмі aSc Розклади можна вказати, що заняття проводяться в декількох будівлях і оптимізувати перехід вчителів між будівлями
- Повністю настроюється. Програма може бути адаптована до конкретних потреб кожного навчального закладу
- aSc Заміни. Корисний модуль програми, який дозволяє планувати заміни вчителів, їх роздрукувати і відправляти повідомлення про зміни в розкладах. Водночас заміни можуть вводити кілька користувачів.
- Сервер EduPage. На цьому веб-сервері, використовуючи aSc Розклади в Інтернеті, Ви можете швидко створити привабливий сайт навчального закладу і одним натисненням в програмі aSc Розклади в нього завантажити розклади, заміни, опублікувати іншу актуальну інформацію
- Настроюється під самі різні навчальні заклади. Програмою aSc Розклади користуються майже у всіх країнах світу, вона розробляється так, щоб змогла задовольнити найрізноманітніші потреби навчальних закладів
- aSc Розклади в Інтернеті. Це модуль програми, який дозволяє створити веб-сайт школи і публікувати розклади в Інтернеті
- Електронний щоденник. Використовуючи створений розклад, можете почати користуватися і електронним щоденником. Вчителі можуть підключитися до нього, ввести навчальні програми, відвідуваність і багато іншого

Недоліки

- Рішення платне.
- Дуже складний функціонал для початківця.
- Програмний продукт йде як клієнт на комп’ютер, було би краще реалізувати в формі веб-сервісу.

1.2.3 Timetableweb

Timetable Web, онлайн сервіс для автоматичної генерації шкільних розкладів. Всі операції виконуються в веб браузері, від введення даних до кінцевого друку розкладу в форматі PDF, ви можете використовувати будь-який пристрій щоб скористатися цим сервісом.

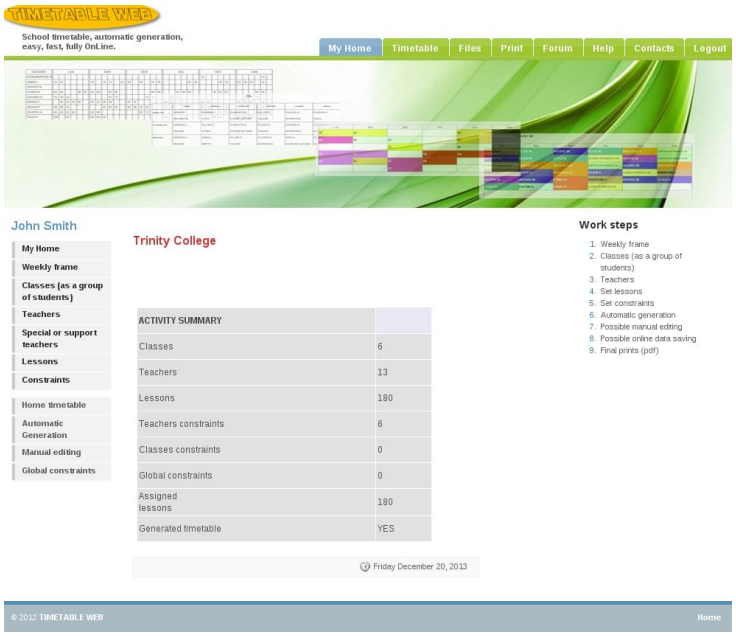


Рис. 1.6 Головна сторінка

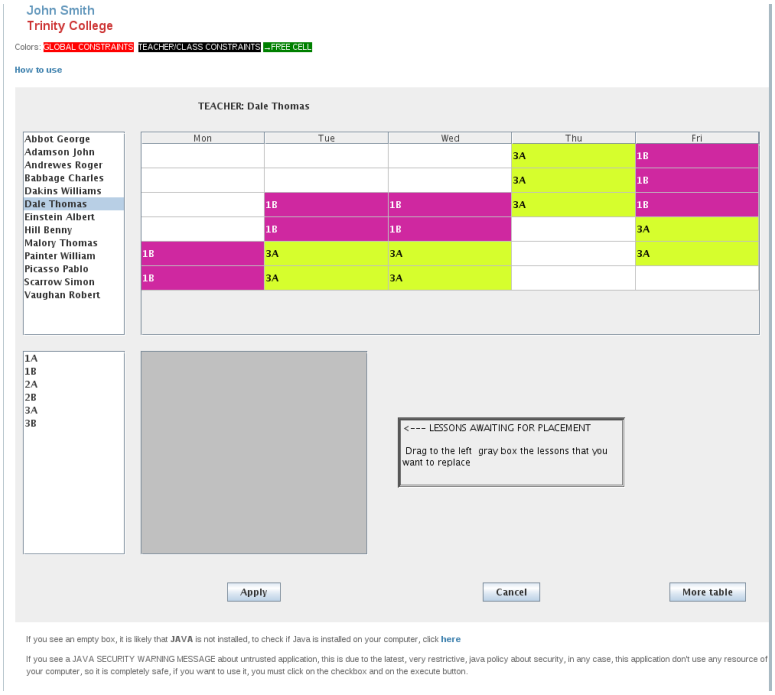


Рис. 1.7 Інтерфейс вводу параметрів розкладу

Переваги

- Повністю онлайн
- Установка не вимагається
- Всі операційні системи
- Автоматична генерація протягом декількох секунд
- Ручне редагування з графічним інтерфейсом
- Pdf друк
- Налагодження розкладу графічним інтерфейсом
- Роздільні-класи
- Форум підтримки
- Періодичне резервне копіювання даних

Недоліки

- Призначений тільки для планування розкладів для шкіл
- Не має можливості налагодити розклад більше ніж на тиждень

1.3 Висновки

В даному розділі наведено аналітичний огляд існуючих рішень. Багато з них мають суттєві переваги та багатий функціонал, проте жодна з існуючих систем не орієнтована на врахування індивідуального побажань користувачів при формуванні навчального плану. Тому виникає необхідність розробки такої системи, яка б забезпечила інтеграцію з найпопулярнішими рішеннями в області планування власних справ.

Також наведена класифікація основних задач, що вирішуються в рамках теорії розкладів. В даній роботі розглядається задача багатокритеріальної оптимізації календарного плану навчального процесу.

2. ВИМОГИ ДО ХАРАКТЕРИСТИК ОБ'ЄКТУ ПРОЕКТУВАННЯ

N = {1,…,n} – множина робіт на один семестр. Завдання виконуються послідовно одним студентом (*з перериваннями або без, Machine Scheduling Problem*).

Кожна робота характеризується наступними параметрами:

pj > 0 – тривалість виконання j-ої роботи.

Dj > 0 – крайній термін здачі j-ої роботи.

Wj > 0 – вага j-ої роботи (залежить від типу предмета: залік, диференціальний залік, екзамен та від типу роботи: лабораторна робота, реферат, розрахункова робота, курсова робота, дипломна робота і т.д.)

rj >= 0 – час отримання роботи (не всі роботи видаються в 0-ий момент часу, видача деякий робіт відбувається протягом семестру)

Sj > 0 – момент початку (start) виконання j-ої роботи (величина, яку необхідно визначити для кожної роботи).

Sj > rj – роботу можна починати виконувати тільки після її отримання.

Zjk, де k = 1,..,m – множина моментів часу до k-ої можливості здачі j-ої роботи.

Момент завершення виконання роботи (hard stop критерій):

Cj = Sj + pj + zj1 <= Dj.

Якщо

Sj < Si,

то Sj + pj <= Si, где i != j (роботи не можуть виконуватися паралельно).

Задача полягає в оптимізації суми штрафних функцій (сума дат завершення кожної роботи).

$$\sum_{j}^{n} Cj \rightarrow min \quad (2.1)$$

Ще одна величина, яка підлягає оптимізації – це показник рівномірності навантажень на кожен робочий день (Ni).

2.1 Перелік обов’язкових (Hard Stop) критеріїв

- Кожна робота повинна бути закінчена і здана до крайнього строку (deadline).
- Виконану роботу можна здати лише в той день, коли є пара з викладачем, який приймає роботу.
- Почати виконання роботи можливо за умови, якщо немає інших справ в цей час.

2.2 Перелік необов'язкових (Soft Stop) критеріїв

- Обов’язкові задачі з високим пріорітетом повинні виконуватися якомога раніше.
- Рівномірність завантаженості кожного робочого дня (показник завантаженості не повинен перевищувати допустиму норму).
- Сумма дат завершення робіт мінімальна. Ця умова відображає бажання студента виконати всі роботи якомога швидше.

2.3 Об’єктна модель предметної області

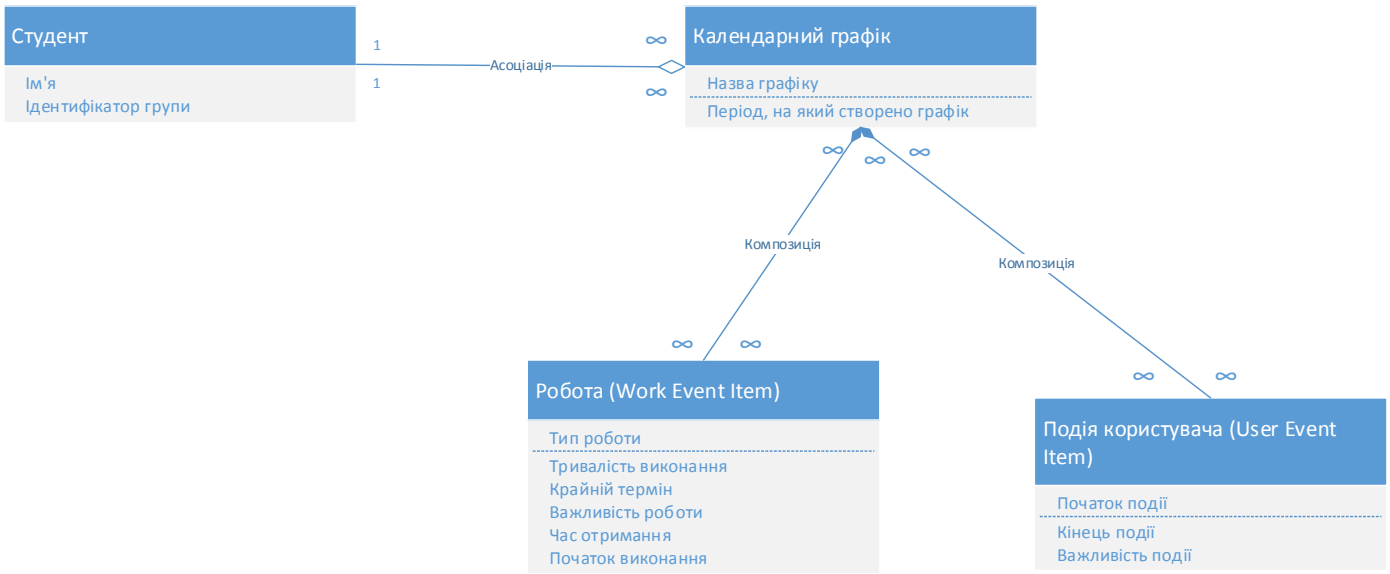


Рис.2.1 Діаграма об’єктів та їх взаємозв’язків (ERD)

Календарний графік включає множину робіт (Work Event Item), яка має наступні атрибути:

- Тип роботи (лабораторна робота, розрахункова робота, додаткова робота, курсова робота, дипломна робота, магістерська робота).
- Крайній термін – дата, до якої необхідно виконати та здати роботу.
- Важливість роботи або пріоритет (низький, середній, високий).
- Початок виконання – величина, значення якої необхідно визначити при формуванні оптимального календарного графіку.

Календарний графік включає множину подій користувача (User Event Item). До таких подій належать ті, які відносяться до власних справ користувача (пов'язані з його працевлаштуванням або дозвіллям).

2.4 Функціональні аспекти системи. Діаграма прецедентів

Діаграма прецедентів є графом, що складається з множини акторів, прецедентів (варіантів використання) обмежених границею системи (прямокутних), асоціації між акторами та прецедентами, відношення серед прецедентів та відношень узагальнення між акторами. Такі діаграми відображають елементи моделі варіантів використання.

Рис2.2 Діаграма прецедентів

Ролі в системі:

- Студент.
- Староста групи.
- Адміністратор.

Кожний користувач повинен пройти етап автентифікації та авторизації перед використанням системи задля підтвердження особистості та отримання відповідних прав доступу.

Студент може додавати власні роботи (наприклад додаткові роботи) до вже існуючих робіт для всієї групи. На основі переліку всіх запланованих на 1 семестр робіт та власного календарного графіку студент має можливість отримати оптимальний календарний план виконання робіт.

Староста групи також є студентом та має змогу скористатися всіма його можливостями та правами. Також додатково може заповнювати роботи на семестр для всієї групи, які автоматично з’являтимуться в календарному плані кожного студента даної групи.

Адміністратор має доступ до статистичних даних (середній час виконання певного типу роботи в групі, середня завантаженість групи та інші).

2.5 Структурні особливості взаємодії об’єктів. Діаграма послідовностей.

Діаграма послідовності – в UML, відображає взаємодії об’єктів, підпорядкованих за часом. Зокрема, такі діаграми відображають задіяні сутності та послідовність відправлених ними повідомлень.

Діаграма послідовності взаємодії старости групи з системою:

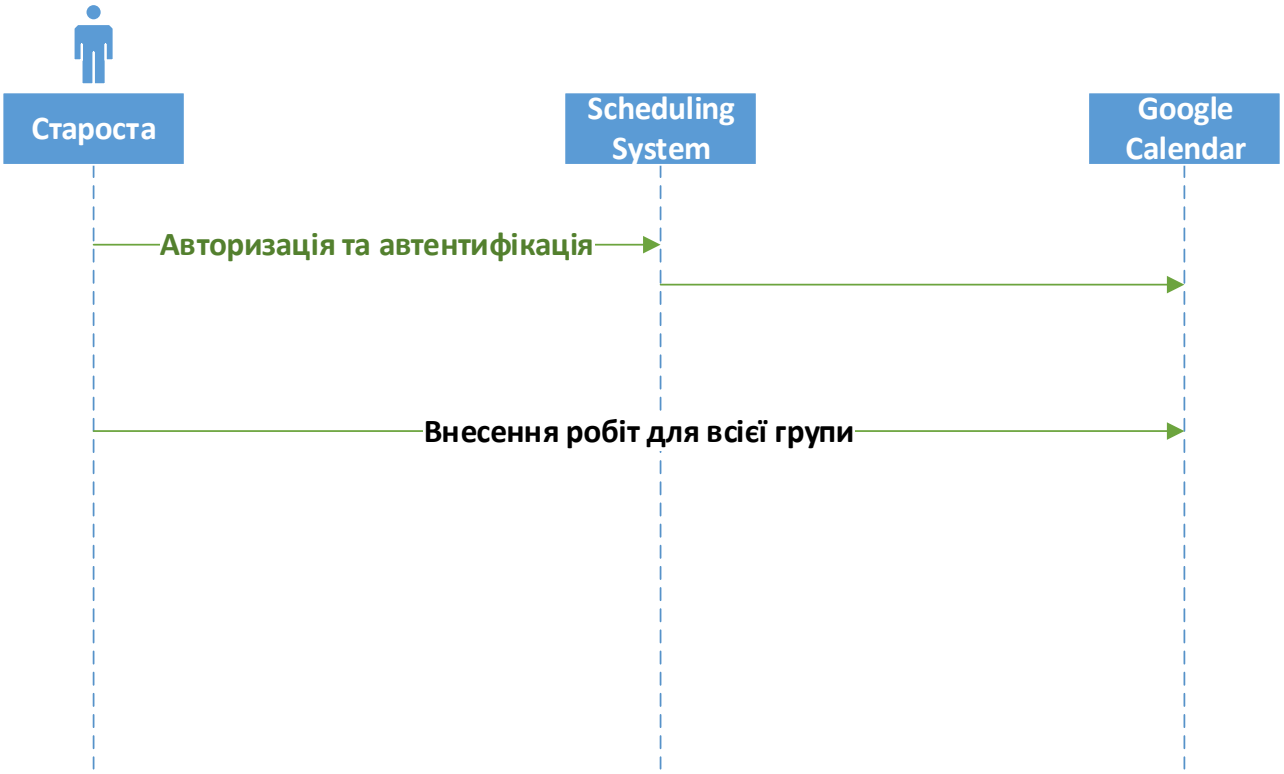


Рис.2.3 Взаємодія старости групи з системою

Діаграма послідовності взаємодії студента з системою:

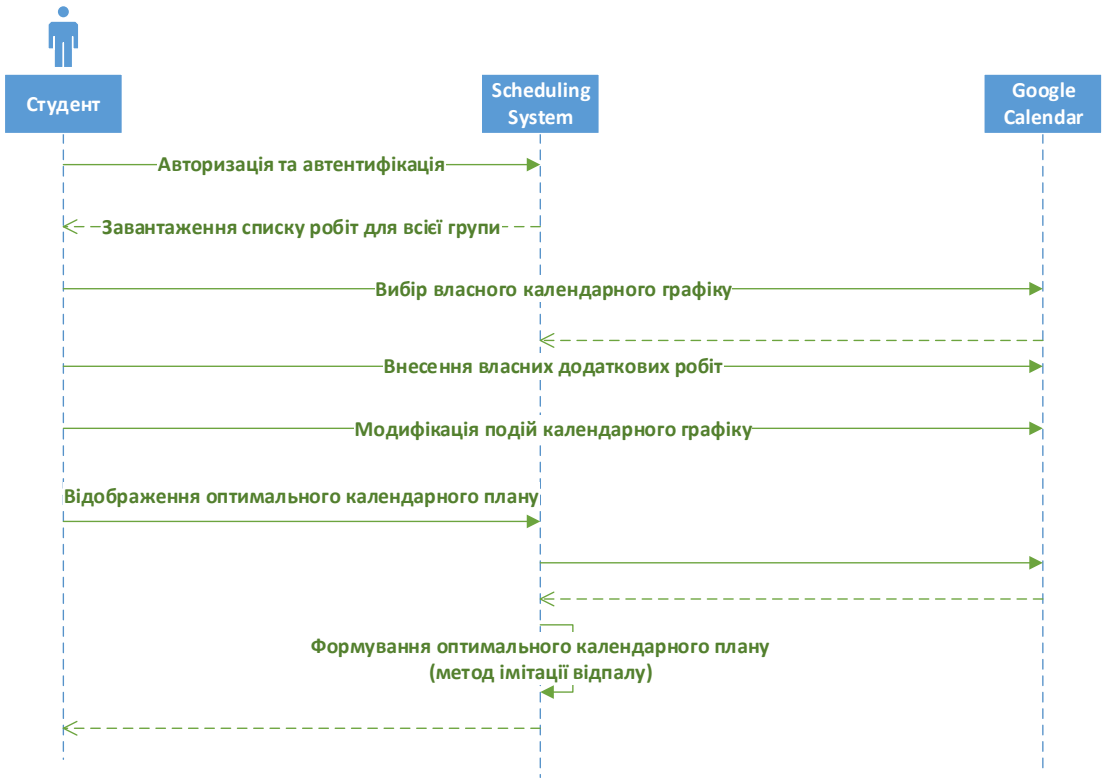


Рис.2.4 Взаємодія студента з системою

Діаграма послідовності взаємодії адміністратора з системою:

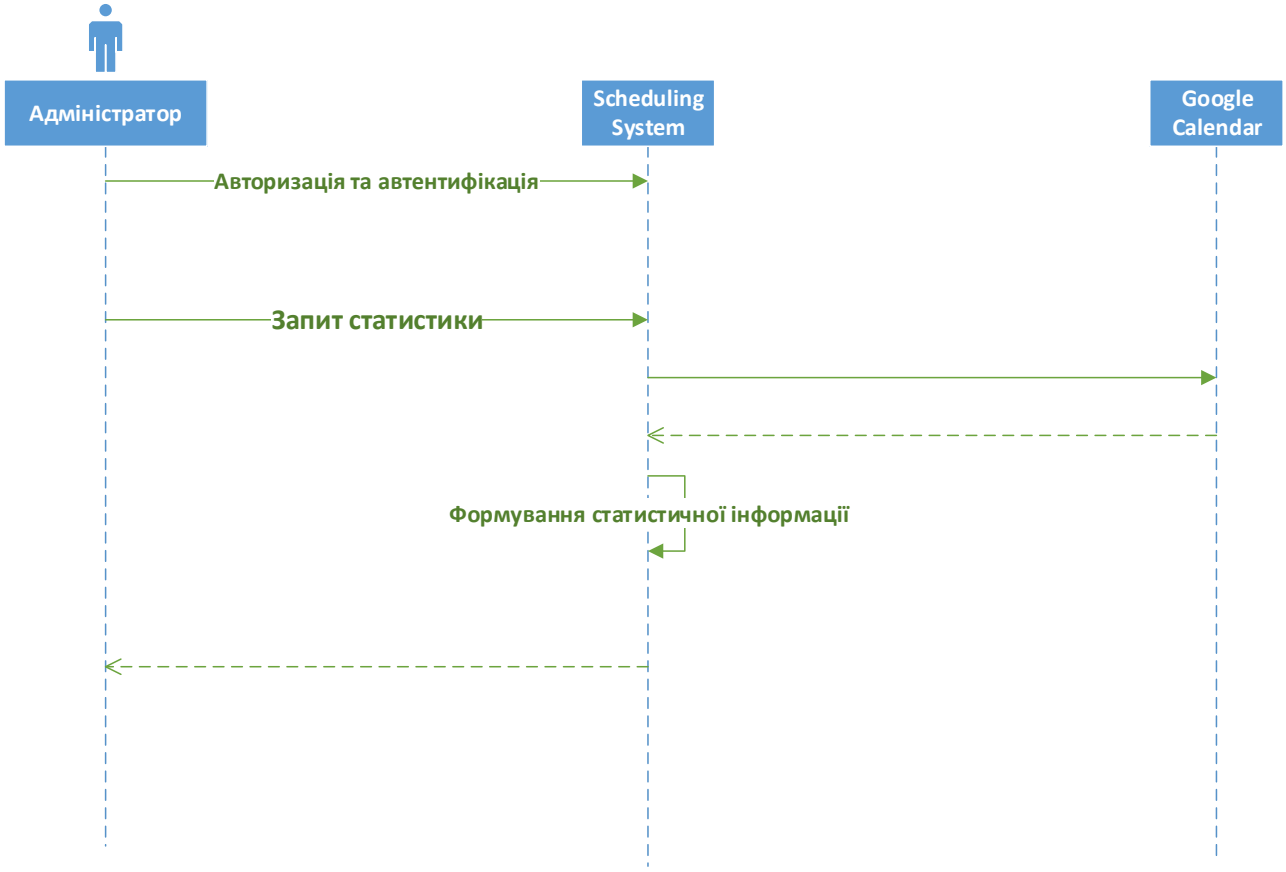


Рис.2.5 Взаємодія адміністратора з системою

2.6 Опис потоків даних. Діаграма потоків даних (Data Flow Diagram)

Data Flow – методологія графічного структурного аналізу, що описує зовнішні по відношенню до системи джерела та адресати даних, логічні функції та сховища даних, до яких здійснюється доступ. Представляє собою ієрархію функціональних процесів, що пов’язані між собою потоками даних.

Діаграма потоків даних (Data Flow diagram, DFD) – один з основних інструментів структурного аналізу і проектування інформаційних систем.

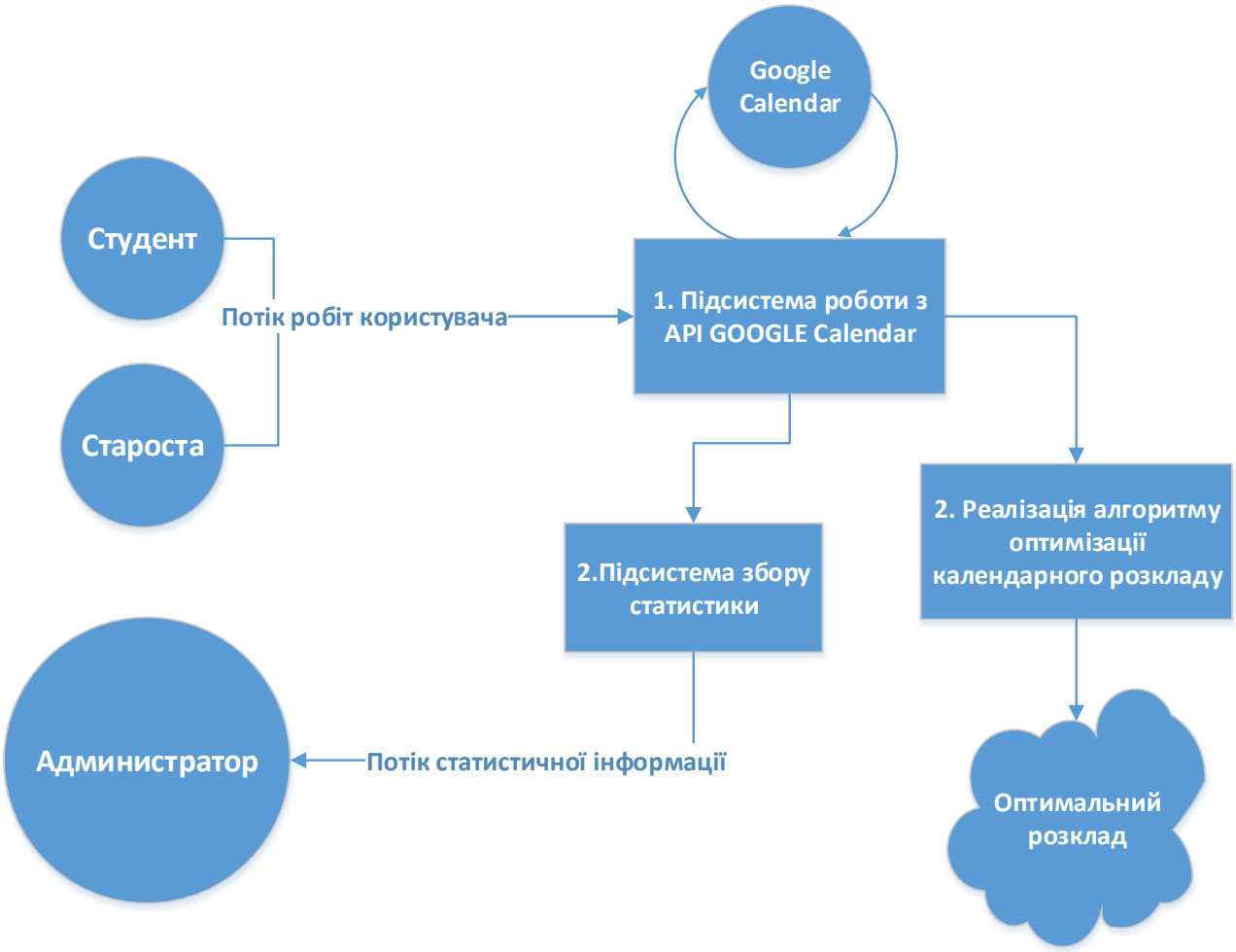


Рис.2.6 Data Flow diagram

2.7 Висновки

В даному розділі було наведено математичний опис предметної області. Була проведена об'єктна декомпозиція та представлення взаємозв’язків між сутностями предметної області результатом якої є ERD діаграми. Створена діаграма прецедентів, що відображає основні функціональні аспекти системи та відносини розширення між акторами системи. Діаграми послідовностей представляють порядок взаємодії сутностей системи та їх обмін повідомленнями. Проведено графічний структурний аналіз з використанням Data Flow методології, що дозволило виділити основні потоки даних в інформаційній системі та функціональні елементи, що виконують їх трансформації.

3. МАТЕМАТИЧНА МОДЕЛЬ ТА ОСОБЛИВОСТІ РЕАЛІЗАЦІЇ АЛГОРИТМІВ ОПТИМІЗАЦІЇ КАЛЕНДАРНОГО ПЛАНУ

3.1 Метод імітації відпалу

За основу даного методу було взято процес кристалізації речовини, який використовується металургами для підвищення однорідності металу. У кожного металу є кристалічна решітка, що описує геометричне положення атомів речовини. Сукупність позицій усіх атомів називається *станом системи*, а кожному стану відповідає певний рівень енергії. Ціль відпалу – привести систему в стан із найменшою енергією. Чим нижчий рівень енергії, тим “краще” кристалічна решітка, тобто у неї менше дефектів і щільніший метал.

В ході відпалу метал спочатку нагрівають до деякої температури, що змушує атоми кристалічної решітки покинути свої позиції. Потім починається повільне і контрольоване охолодження. Атоми прагнуть потрапити в стан з меншою енергією, проте, з певною ймовірністю вони можуть перейти і в стан з більшою енергією. Ця ймовірність зменшується разом з температурою. Перехід в гірший стан допомагає в результаті знайти стан з енергією меншою, ніж початкова. Стійка кристалічна решітка відповідає мінімуму енергії атомів, тому він або переходить в стан з меншим рівнем енергії, або залишається на місці. (Цей алгоритм називають також алгоритмом Н. Метрополіса, за іменем його автора). У випадку математичної задачі роль частинок речовини виконують параметри, а роль енергії – функція, що характеризує оптимальність набору даних параметрів.

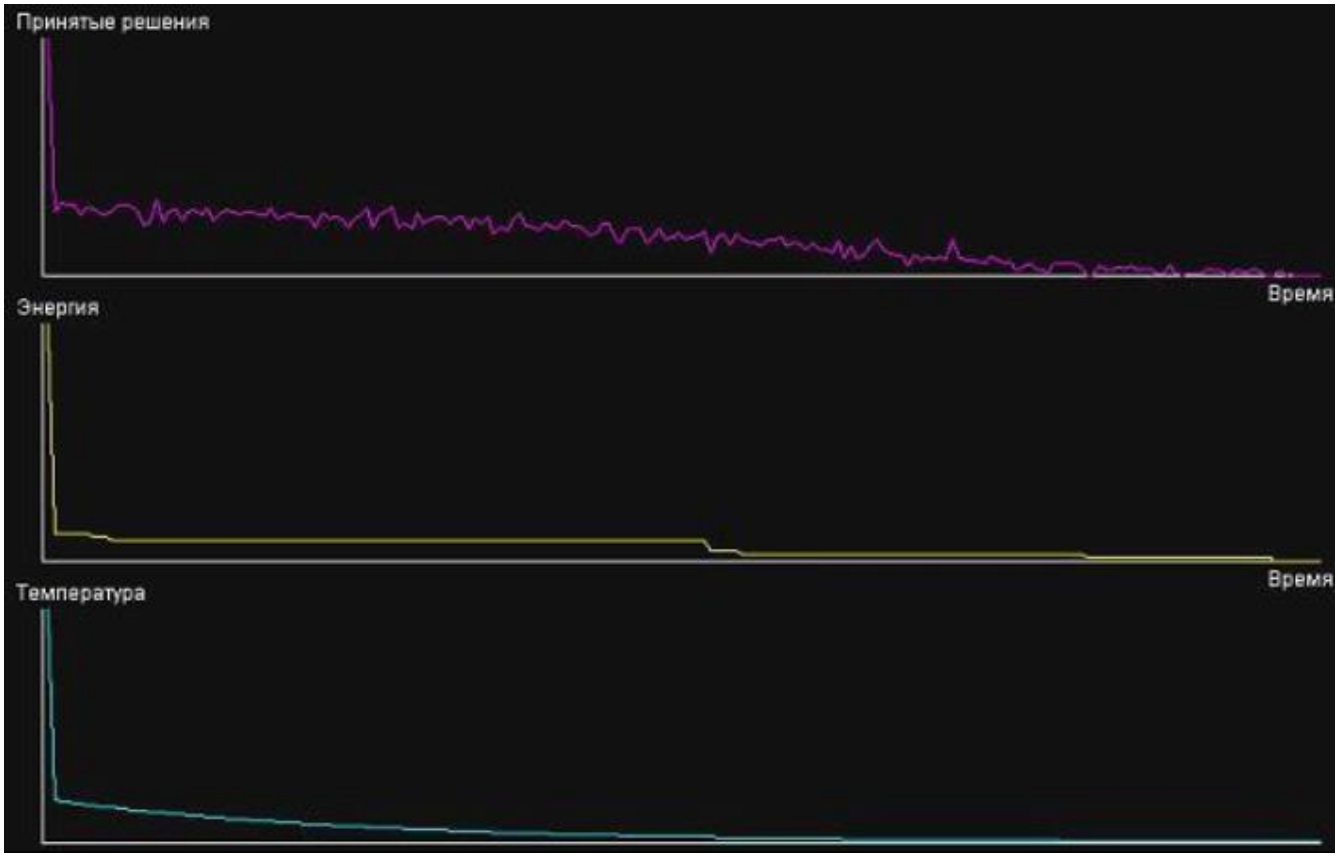


Рис.3.1 Приклад результатів задачі розташування фігур на шахматній дошці з вик. методу імітації відпалу [7]

Фактично, при моделюванні необхідно знайти деяку точку (або множину точок), на якій досягається мінімум деякої числової функції $F(\vec{x})$. Будується послідовність точок $\vec{x}_0, \vec{x}_1, \dots, \vec{x}_n$, де $\vec{x}_0$ відповідає початковому розподілу. При досягненні точки $\vec{x}_n$ алгоритм завершує свою роботу. Нехай розглядається поточна точка $\vec{x}_i$. До неї застосовується деякий оператор А, який довільним чином модифікує дану точку, в результаті чого отримуємо нову точку $\vec{x}_i{}^*$. Ймовірність, з якою $\vec{x}_i{}^*$ стане наступною точкою $\vec{x}_{i+1}$ рівна $P\left(\vec{x}_i{}^*, \vec{x}_{i+1}\right)$, де P – розподіл Ґіббса:

$$P\left(\vec{x}_i{}^* \to \vec{x}_{i+1} \middle| \vec{x}_i\right) = \begin{cases} 1, & F\left(\vec{x}_i{}^*\right) - F(\vec{x}_i) < 0 \\ \exp\left(-\dfrac{F\left(\vec{x}_i{}^*\right) - F(\vec{x}_i)}{T_i}\right), & F\left(\vec{x}_i{}^*\right) - F(\vec{x}_i) \geq 0 \end{cases} \quad (3.1)$$

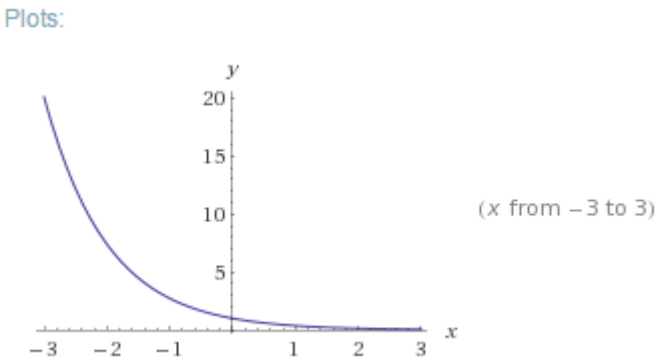


Рис.3.2 $y = e^{-x}$, де $x = \frac{F(\vec{x}_i^{\,*}) - F(\vec{x}_i)}{T_i}$

$T_i > 0$ – елементи довільної спадаючої послідовності, що сходиться до нуля. Ця послідовність представляє собою аналог температури, що знижується під час реального фізичного процесу. Швидкість і закон спадання можуть бути заданими автором алгоритму будь-яким чином.

Алгоритм імітації відпалу можна продемонструвати у вигляді наступної блок-схеми:

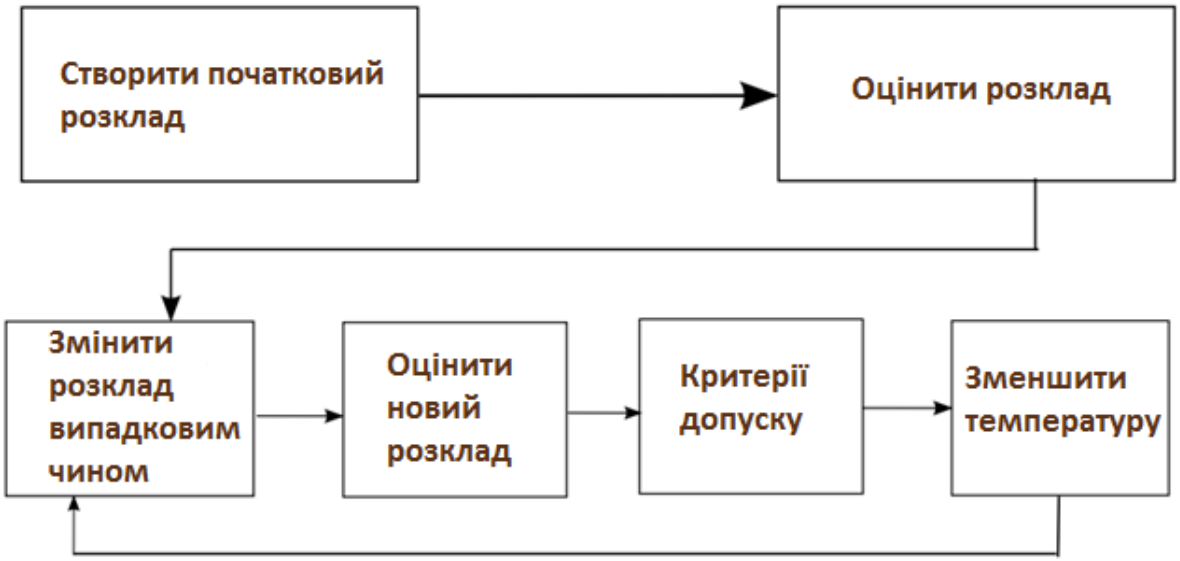


Рис.3.3 Блок схема алгоритму імітації відпалу

Алгоритм імітації відпалу схожий на градієнтний спуск, але за рахунок випадковості вибору проміжної точки потраплятиме в локальні мінімуми рідше за градієнтний спуск. Даний алгоритм не гарантує знаходження мінімуму

функції, але при правильній генерації випадкової точки в просторі X, як правило відбувається покращення початкового наближення.

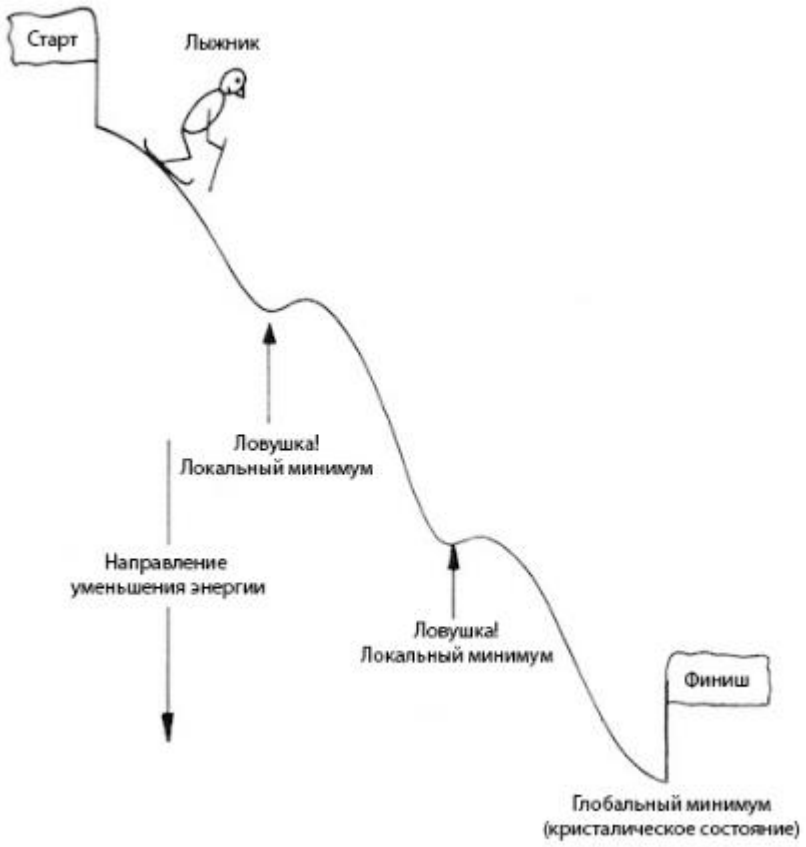


Рис.3.4 Процес пошуку глобального мінімуму [8]

Розглянемо основні кроки алгоритму:

1) Генерація початкового розкладу $Schedule_0$.

Початковий розклад повинен задовольняти наступні hard stop критерії:

a) Кожна робота повинна бути запланованою на дату не раніше дати видачі та не пізніше дати здачі.

$$r_i \leq S_i + d_i \leq D_i, i = 0, \dots n \quad (3.2)$$

b) Завантаженість кожного робочого дня не має перевищувати кількості вільного часу студента в даний день (або параметр, що визначає кількість годин, яка виділяється на навчання, якщо такий вказаний користувачем при заповненні задач на семестр)

$$freeTime_k - \sum_{i=0}^{n} p_i \geq 0, \qquad \text{де } i = 0, \dots n \quad (3.3)$$

k – номер дня, $\sum_{i=0}^{n} p_i -$ кількість годин в день, що виділяється для виконання на навчання.

2) Розрахунок функції енергії для початкового розкладу.

Розглянемо логіку формування даної функції. Кожен студент намагається здати лабораторні роботи якомого раніше, адже зрив строків здачі навчального плану карається зняттям балів з виконаної роботи, неатестованими заліками, а також, що найгірше, виключенням з університету за неуспішність у навчанні. Своєчасна здача робіт забезпечує шанс отримати заохочувальні або додаткові бали, а також можливість перездати роботу, у випадку, коли у викладача є багато зауважень, що знижують фінальну оцінку. Тому важливим є оптимізації дат завершення кожної роботи за увесь семестр. Чим швидше будуть закриті всі роботи за семестр – тим більше часу потім маємо на підготовку розрахункових, курсових робіт та на підготовку до залікового тижня та сесії. З наведених міркувань функція енергії має прийняти наступний вигляд:

$$F(Schedule_i) = \sum_{i=0}^{n} C_i \ \rightarrow min, \qquad (3.4)$$

Де C_i – дата завершення виконання і-ої задачі.

Оскільки на початку кожного семестру кожен студент визначає для себе пріоритет того чи іншого предмету та виділяє більше часу на засвоєння більш важливих для нього предметів, то необхідно корегувати важливість менш пріоритетних задач, щоб для них залишився час. Проте виконання більш пріоритетних завдань повинно плануватися раніше, від менш пріоритетних. З даною метою було введено наступну двоступеневу систему ваги кожної задачі:
За особистим пріоритетом предмета для студента (*PersonalSubjectPriority*):

- Високий пріоритет (High) – 5.
- Середній пріоритет (Medium) – 3.
- Низький пріоритет (Low) – 1.

За типом фінальної здачі даної дисципліни (*CompletionTypePriority*):

- Екзамен – 5.

- Диференційований залік – 3.
- Звичайний залік – 1.

Для розрахунку фінального значення ваги предмета використовується наступна формула:

$$w_i = PersonalSubjectPriority_i * CompletionTypePriority_i, i = 0, \dots n$$

де w_i знаходиться в діапазоні $[1, 25]$.

З врахуванням двоступеневої системи пріоритетів функція енергії набуває наступного вигляду:

$$F(Schedule_i) = \sum_{i=0}^{n} C_i * w_i \quad (3.5)$$

Тобто задачі з більшою вагою повинні виконуватися раніше.

Оскільки нормування свого робочого дня та виділення часу на відпочинок є основною метою планування календарного розкладу для кожного студента, то необхідно враховувати наступні критерії:

a) Навантаження кожного робочого дня не перевищує певної норми (даний параметр може налаштовуватись в системі, але рекомендоване значення – 6 годин)
b) Навантаження кожного робочого дня не перевищує значення параметру, що відповідає за кількість годин, які виділяються студентом на навчання, якщо такий параметр вказано.

Таким чином важливою задачею є мінімізація середнього значення завантаженості робочого дня. Функція енергії прийме наступний вигляд:

$$F(Schedule_i) = \sum_{i=0}^{n} C_i * w_i * \frac{\sum_{k=0}^{s} \frac{freeTime_k - sum_k}{freeTime_k}}{s} \quad (3.6)$$

Де s – кількість робочих днів в семестрі, k – номер робочого дня,

$sum_k = \sum_{i=0}^{n} p_{ik}$ – сума тривалості кожної роботи, що запланована на k-ий день.

3) Генерації нового розкладу на базі існуючого. Даний процес відбувається наступним чином:

a) Обирається випадковий день, в складі якого є задачі.
b) Обирається випадкова задача в складі обраного дня.
c) Обрана задача переміщується до випадкового дня з врахуванням hard-stop критеріїв.

4) Розрахувати значення функції енергії для нового розкладу.
5) Порівняти значення функції енергії для теперішнього розкладу та створеного
   a) Якщо $F(Schedule_{i+1}) \leq F(Schedule_i)$ – тоді обрати i+1 розклад за поточний варіант.
   b) Якщо $F(Schedule_{i+1}) > F(Schedule_i)$, тоді розрахувати ймовірність прийняття рішення за допомогою функції розподілу Гібса.
6) Умовою припинення виконання даної послідовності кроків є охолодження системи та її перехід в стабільне положення.

$$T \leq 1 \quad (3.7)$$

Розглянемо функцію температури, вона має наступний вигляд:

$$T_{i+1} = T_0 * 0.9^i, T_0 = 10000^o \quad (3.8)$$

Для сповільнення швидкості спадання температури використовувалась також наступна функція температури:

$$T_{i+1} = T_0 * 0.9^{\sqrt{i}} \quad (3.9)$$

3.2 Жадібний алгоритм

Жадібний алгоритм (Greedy algorithm) – алгоритм, що полягає в прийнятті локально оптимальних рішень на кожному етапі, припускаючи, що кінцеве рішення виявиться оптимальним. Жадібні алгоритми не завжди приводять до оптимального рішення, але в багатьох задачах дають необхідний розв’язок.

Процес розробки жадібного алгоритму складається з наступних етапів:

1. Приведення задачі оптимізації до вигляду, коли після виконання вибору залишається вирішити тільки одну підзадачу.
2. Довести, що завжди існує таке оптимальне рішення вихідної задачі, яке можна отримати шляхом жадібного вибору, таким чином, що такий вибір завжди є можливим.
3. Продемонструвати оптимальну структуру, показавши, що після жадібного вибору залишається підзадача, що має властивість, що після об'єднання оптимального рішення підзадачі зі здійсненим жадібним вибором приводить до оптимального рішення вихідної задачі.

В основі кожного жадібного алгоритму майже завжди знаходиться більш складне рішення в стилі динамічного програмування. Для того, щоб визначити чи здатен даний тип алгоритму отримати ефективне рішення, можна виділити два ключових компоненти – властивість жадібного вибору і оптимальну підструктуру. Якщо вдається продемонструвати, що задача володіє обома властивостями, то з великою ймовірністю для неї можна розробити жадібний алгоритм.

Властивість жадібного вибору – глобальне оптимальне рішення можна отримати шляхом прийняття локально оптимальних рішень.

Оптимальна підструктура – якщо в оптимальному рішенні задачі містяться оптимальні рішення її підзадач, тоді задача демонструє дану властивість.

Розглянемо основні кроки алгоритму:

Розглянемо основні кроки алгоритму:

1) Генерація початкового розкладу $Schedule_0$.

Початковий розклад повинен задовольняти наступні hard stop критерії:

c) Кожна робота повинна бути запланованою на дату не раніше дати видачі та не пізніше дати здачі.

$$r_i \leq S_i + d_i \leq D_i, i = 0, \dots n \quad (3.10)$$

d) Завантаженість кожного робочого дня не має перевищувати кількості вільного часу студента в даний день (або параметр, що

визначає кількість годин, яка виділяється на навчання, якщо такий вказаний користувачем при заповненні задач на семестр)

$$freeTime_k - \sum_{i=0}^{n} p_i \geq 0, \qquad \text{де } i = 0, \dots n \quad (3.11)$$

k – номер дня, $\sum_{i=0}^{n} p_i -$ кількість годин в день, що виділяється для виконання на навчання.

2) Розрахунок значення цільової функції для початкового розкладу:

$$F(Schedule_0) = \sum_{i=0}^{n} C_i * w_i * \frac{\sum_{k=0}^{s} \frac{freeTime_k - sum_k}{freeTime_k}}{s} \quad (3.12)$$

3) Генерація нового розкладу шляхом обміну обраної задачі в інший робочий день, що задовольняє всі hard-stop критерії.
4) Розрахунок значення цільової функції для нового розкладу.
5) Порівняння значень цільових функцій початкового і згенерованого розкладів:
   a) Якщо $F(Schedule_0) \leq F(Schedule_i)$, тоді обираємо $Schedule_i$ як поточне оптимальне рішення за принципом жадібного вибору (локально обираємо найкраще рішення з метою отримання остаточного результату, який приводить до глобального мінімуму).
   b) Якщо $F(Schedule_0) > F(Schedule_i)$, тоді початковий розклад залишається поточним оптимальним рішенням.
   c) Повторюємо кроки обмежену кількість ітерацій.

3.3 Висновки

В даному розділі були описані особливості реалізації та математична модель алгоритму імітації відпалу та жадібного алгоритму. Розглянуто основні принципи ймовірнісного прийняття рішень з використанням розподілу Гібса, покрокове формування штрафної функції. Розроблено дворівневу систему розрахування ваги кожного завдання з метою врахування особистого

пріоритету користувача та важливості предмету з точки зору навчального плану. Наведено опис процесу розробки алгоритму на основі принципу жадібного вибору та властивості оптимальної підструктури задач.

4. ВИБІР ТЕХНОЛОГІЙ ДЛЯ РЕАЛІЗАЦІЇ ПРОТОТИПУ

4.1 Spring Framework

Платформа Spring - популярна платформа додатків з відкритим кодом, призначена для спрощення розробки для J2EE. Вона складається з контейнера, платформи управління елементами і набору інтегрованих служб для веб-інтерфейсів користувача, транзакцій і збереження стану. До складу платформи Spring Spring Web Входить MVC - розширювана платформа MVC для створення веб-додатків.

Незважаючи на те, що Spring Framework не забезпечував якусь конкретну модель програмування, він став широко поширеним в Java-співтоваристві головним чином як альтернатива і заміна моделі Enterprise JavaBeans. Spring Framework надає велику свободу Java-розробникам в проектуванні; крім того, він надає добре документовані і легкі у використанні засоби вирішення проблем, що виникають при створенні додатків корпоративного масштабу.

Spring Framework забезпечує вирішення багатьох завдань, з якими стикаються Java-розробники і організації, які хочуть створити інформаційну систему, засновану на платформі Java. Через широку функціональності важко визначити найбільш значущі структурні елементи, з яких він складається. Spring Framework НЕ повністю пов'язані з платформою Java Enterprise, незважаючи на його масштабну інтеграцію з нею, що є важливою причиною його популярності.

Spring Framework, ймовірно, найбільш відомий як джерело розширень (функції), потрібних для ефективної розробки складних бізнес-додатків поза великовагових програмних моделей, які історично були домінуючими в промисловості. Ще одне його достоїнство в тому, що він ввів раніше невикористовувані функціональні можливості в сьогоднішні панівні методи розробки, навіть поза платформи Java.

Цей фреймворк пропонує послідовну модель і робить її придатною до більшості типів додатків, які вже створені на основі платформи Java. Вважається, що Spring Frameworkреалізует модель Розробки, засновану на кращих стандартах індустрії, і робить її доступною в багатьох областях Java.

Основна перевага Spring - можливість розробки програми як набору слабозв'язаних компонентів. Чим менше компоненти програми знають один про одного, тим простіше розробляти новий і підтримувати існуючий функціонал програми. Класичний приклад - управління транзакціями. Spring дозволяє вам керувати транзакціями абсолютно незалежно від основної логіки взаємодії з БД. Зміна цієї логіки не порушить транзакційність, так само як зміна логіки управління транзакціями не зламає логіку програми. Spring модульність заохочує. Компоненти можна додавати і видаляти (майже) незалежно один від одного. В принципі, додаток можна розробити таким чином, що воно навіть не буде знати, що управляється Spring. Також Spring помітно спрощує модульне тестування (модульного тестування): в компонент, розроблений для роботи в IoC контейнері дуже легко створювати тестові залежності і перевірити роботу тільки цього компонента. Ну, і як приємне доповнення, весна сильно полегшує ініціалізацію і налаштування компонентів додатка, дозволяючи гнучко налаштовувати додаток без істотних змін Java-коду.

Переваги слабкої зв'язаності компонентів:

1. спрощення ініціалізації і налаштування компонентів,
2. спрощення модульного тестування,
3. спрощення розробки та підтримки програми в цілому.

Spring був розроблений як альтернатива EJB від самого початку, і пропонує багато можливостей, що полегшують розробку. Використання цього фреймворку значно збільшить якість розробки.

4.2 MVC

Принцип **MVC** у веб-програмуванні (Model - View - Controller, Модель - Подання (Вид) - Контролер) - одна з найбільш вдалих ідей на сьогоднішній день. Принцип MVC інтуїтивно зрозумілий на перший погляд, але не дуже простий при поглибленні. Спочатку розглянемо, для чого він призначений.

Принцип MVC, дозволяє розділити реалізацію логіки докладання, зовнішній вигляд (графічний інтерфейс, GUI) і взаємодія з користувачем.

Це призводить до більш структурованого коду, дозволяє працювати над проектом більш спеціалізованим людям, спрощує підтримку коду, робить його більш логічним і зрозумілим. Зміна в одному з компонентів мінімально впливає на інші. Можна до однієї моделі підключати різні види, різні контролери.

Розглянемо докладніше компоненти.

Model (Модель) - містить т.зв. "Бізнес-логіку" - обробку і верифікацію даних, звернення до баз даних, представляє внутрішній устрій системи. Модель не повинна безпосередньо взаємодіяти з користувачем.

View (Вид, Подання) описує зовнішній вигляд програми.

Controller (Контролер) - сполучна ланка між моделлю і видом, отримує дані від користувача, передає їх моделі, отримує оброблений результат і передає його в представлення.

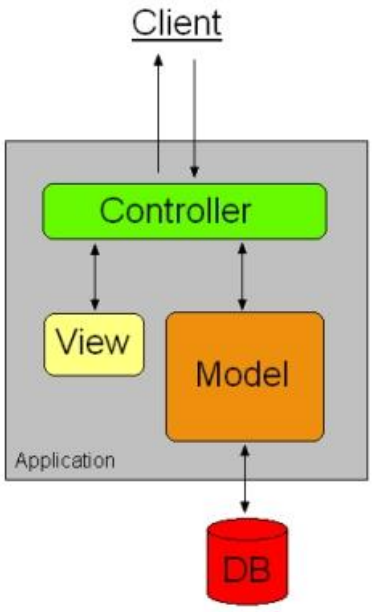


Рис.4.1 Модель MVC [20]

Spring framework надає гарну можливість для реалізації архітектури MVC за допомогою Spring web MVC framework.

Spring web MVC framework забезпечує архітектуру модель-представлення-контролер (MVC) і готові компоненти, які можуть бути використані для розробки гнучких і слабо пов'язаних веб-додатків. Шаблон MVC призводить до поділу різних аспектів застосування (вхідні логіка, бізнес-логіки і логіки UI), забезпечуючи при цьому слабкий зв'язок між цими елементами.

1. **Model** інкапсулює дані додатка і в цілому вони будуть складатися з POJO.
2. **View** відповідає за надання даних моделі і в цілому він генерує HTML, що браузер клієнта може інтерпретувати.
3. **Controller** відповідає за обробку запитів користувачів і побудови відповідної моделі і передає його в уявлення для рендеринга.

DispatcherServlet

Spring Web model-view-controller (MVC) framework розроблений навколо DispatcherServlet, який обробляє всі HTTP-запити і відповіді. Обробка запиту робочого процесу в Spring Web MVC DispatcherServlet показаний на наступній діаграмі:

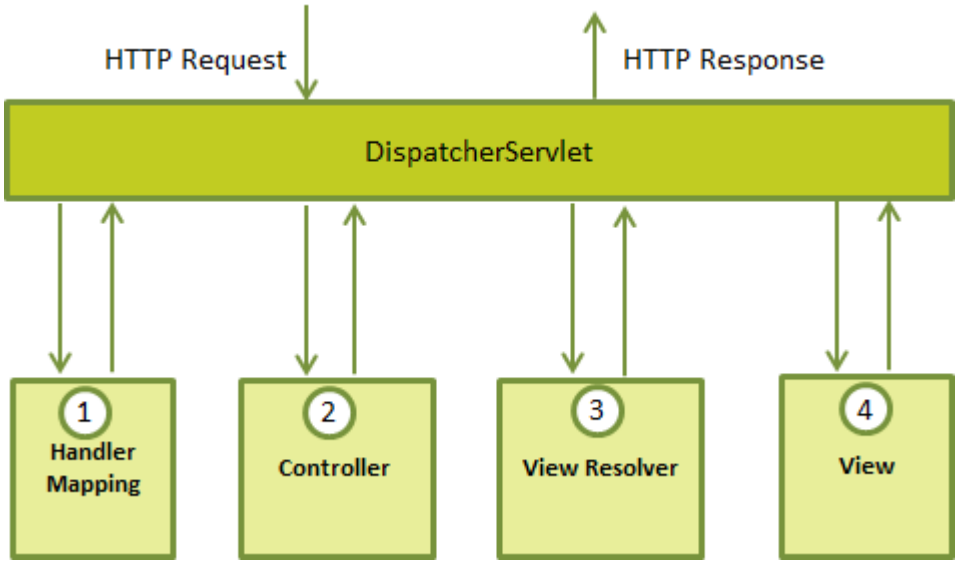


Рис.4.2 Обробка запиту в Spring Web MVC DispatcherServlet [20]

Нижче наводиться послідовність подій, відповідних вхідного запиту HTTP до *DispatcherServlet*:

1. Після отримання запиту HTTP, *DispatcherServlet* консультується *theHandlerMapping* для виклику відповідного *Контролера*.
2. *Контролер* приймає запит і викликає відповідні методи сервісів на основі використовуваного методу GET або POST. Метод служби встановлює дані моделі, засновані на певній бізнес-логіки і повертає ім'я перегляду в *DispatcherServlet*.
3. *DispatcherServlet* буде приймати допомогу від *ViewResolver*.
4. Після того, як перегляд завершено, *DispatcherServlet* передає дані моделі з точки зору, яка, нарешті, винесеному в браузері.

4.3 Авторизація користувача

Користувач має три рівня доступу до додатка: незнайомець, зареєстрований і адміністратор. Для забеспечення даної функціональності потрібно зробити можливість авторизації в ресурсі. Spring framework назає таку можливість за допомогою Spring security.

Spring Security надає комплексні послуги безпеки для J2EE на основі корпоративних програмних додатків. Існує особливий акцент на підтримці проектів, створених з використанням Spring Framework, яка є провідним рішенням J2EE для розробки програмного забезпечення підприємства.

Люди використовують Spring Security з багатьох причин, але більшість з них звертається до проекту після знаходження, що можливостям безпеки J2EE's Servlet Specification або EJB Specification не вистачає глибини, необхідної для типових сценаріїв корпоративних додатків. Незважаючи на те, згадуючи ці стандарти, важливо визнати, що вони не є переносяться на WAR або EAR рівні. Тому, якщо ви переходите до серверних середовищ, як правило, займає багато роботи, щоб змінити конфігурацію безпеки вашого застосування для нового середовища. Використання Spring Security долає ці проблеми, а також приносить вам десятки інших корисних налаштовуюваних функцій безпеки.

Як ви, напевно, знаєте дві основні галузі безпеки додатків є "автентифікація" і "авторизація" (або "контролю доступу"). Це дві основні області, Spring Security. «Автентифікація» являє собою процес встановлення ким користувач є для системи. "Авторизація" відноситься до процесу прийняття рішення чи користувач той за кого себе видає, або чи дозволено виконувати певні дії в вашому додатку.

На рівні автентифікації Spring Security підтримує широкий спектр моделей автентифікації. Більшість з цих моделей автентифікації або надаються третіми особами, або розробляються відповідними органами стандартизації, такими як Engineering Task Force Internet. Крім того, Spring Security надає свій власний набір функцій автентифікації. Зокрема, в даний час Spring Security підтримує інтеграцію автентифікації з усіма цими технологіями:

1. HTTP Basic заголовків перевірки автентичності (стандарт RFC IEFT основі)
2. HTTP Digest заголовки автентифікації (стандартом RFC IEFT основі)
3. HTTP X.509 клієнт обміну сертифікатів (і IEFT RFC на основі стандарт)
4. LDAP (дуже поширений підхід до потреб автентифікації крос-платформних, особливо в великих середовищах)
5. Автентифікація на основі форм (для простих потреб користувальницького інтерфейсу)
6. Автентифікація OpenID
7. Автентифікація на основі заздалегідь встановлених заголовків запиту (наприклад, Computer Associates Siteminder)
8. JA-SIG Центральна служба перевірки справжності (інакше відомий як CAS, який є популярним відкритим вихідним кодом єдиного входу системи)
9. Прозора поширення контексту автентифікації для віддаленого виклику методу (RMI) і HttpInvoker (а протокол Spring Remoting)
10. Автоматичний "пам'ятати мене" тип автентифікації (так що ви можете поставити галочку в полі, щоб уникнути повторної автентифікації протягом заданого періоду часу)

11. Анонімна перевірка справжності (що дозволяє кожен виклик автоматично припускати певну ідентичність безпеки)
12. Run-автентифікацією (що корисно, якщо один виклик повинен приступити до іншої ідентичності безпеки)
13. Java сервіс для автентифікації та авторизації (JAAS)
14. JEE container autentication (так що ви можете використовувати Container Managed Authentication при бажанні)
15. Kerberos
16. Java Open Source Single Sign On (JOSSO)
17. OpenNMS Management Network Platform
18. AppFuse
19. AndroMDA
20. Mule ESB
21. Пряма веб-запиту (ДСР)
22. Grails
23. Tapestry
24. JTrac
25. Jasypt
26. Roller
27. Elastic Path
28. Atlassian Crowd

Багато незалежних постачальників програмного забезпечення (ISV) використовують Spring Security через гнучкий вибір моделей автентифікації. Це дозволяє їм швидко інтегрувати свої рішення з тим, що потрібно їх кінцевим клієнтам, без проведення багатьох змін або вимагання клієнта змінити середовище розробки. Якщо жоден з перерахованих вище механізмів автентифікації не задовольняє ваші потреби, Spring Security є відкритою платформою, і досить просто написати свій власний механізм автентифікації. Багато корпоративних користувачей Spring Security необхідно інтегрувати з "legacy" системами, які не дотримуються будь-яких конкретних стандартів безпеки і Spring Security легко інтегрується з такими системами.

4.4 Spring Data

Місія Spring Data є надання Spring-орієнтованої моделі програмування для доступу до даних, зберігаючи при цьому особливі риси базового сховища даних.

Це дозволяє легко використовувати технології доступу до даних реляційних і не реляційних баз даних, map-reduce фреймворки, і cloud-based сервіси даних. Це батьківський проект, який містить безліч під проектів, які є специфічними для даної бази даних. Проекти розробляються спільно з багатьма компаніями і розробниками [20].

Spring Data JPA не є провайдером JPA. Це бібліотека / структура, яка додає додатковий рівень абстракції, на вершині нашого провайдера JPA. Якщо ми вирішили використовувати Spring Data JPA, repository рівень нашого застосування містить три шари, які описані в наступному:

1. Spring Data JPA забезпечує підтримку для створення JPA репозиторіїв шляхом розширення Spring Data repository інтерфейсів.
2. Spring Data Commons надає інфраструктуру, яка спільно використовується сховищем даних конкретних проектів Spring Data.
3. JPA Provider реалізує API Java Persistence.

На наступному малюнку показана структура нашого repository рівня:

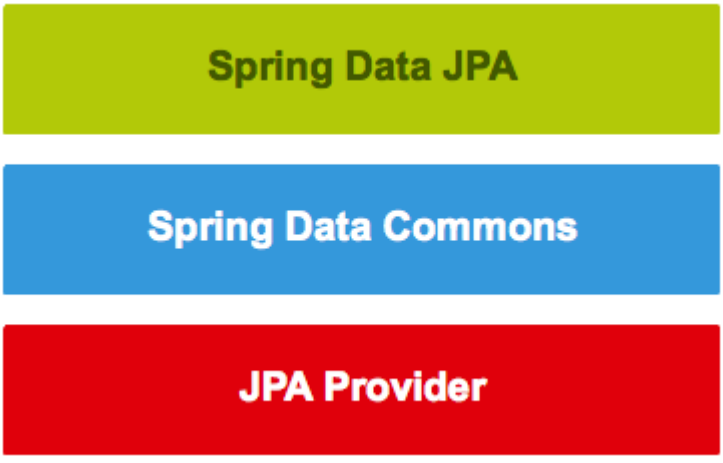


Рис.4.3 Структура repository рівня [20]

На перший погляд здається, що Spring Data JPA робить наш додаток більш складним, і в деякому сенсі це вірно. Це дійсно додає додатковий шар до нашого repository рівня, але в той же час він звільняє нас від написання шаблонного коду.

4.5 Тестування

Для забеспечення цілістності роботи додатка на всіх етапах розробки, гарною практикою є використання тестування. Найкращим рішенням є JUnit.

JUnit є відкритим вихідним фреймворком розробленим з метою написання та запуска тестів в мові програмування Java [24].JUnit, спочатку написаний Ерix Гамма і Кент Бек, зіграв важливу роль в еволюції test-driven development(TDD), який є частиною більшого проекту програмного забезпечення парадигми, відомої як Extreme Programming (XP).

JUnit має графічний користувальницький інтерфейс (GUI), що робить можливим писати і вихідний код тесту швидко і легко. JUnit дозволяє розробнику поступово будувати тестові набори для вимірювання прогресу і виявлення ненавмисних побічних ефектів. Тести можуть працювати безперервно. Результати відразу ж надаються. JUnit показує прогрес тесту в барі, який зазвичай зелений, але стає червоним, коли тест не пройдений. Триваючий список невдалих випробувань з'являється в просторі поблизу нижньої частини вікна дисплея. Кілька тестів можуть бути запущені одночасно. Ні суб'єктивних людських суджень або інтерпретації результатів випробувань не потрібно. Простота JUnit дозволяє розробнику програмного забезпечення легко виправити помилки під час їх виявлення.

Хоча JUnit спочатку була написана для Java, продовженнях розробили для кількох інших мов програмування. Вся сім'я споріднених структур тестування називається XUnit.

4.6 Google calendar API

Мільйони людей використовують Google Calendar, щоб відслідковувати їх події. Calendar API дозволяє інтегрувати додаток з Google Calendar, створюючи нові шляхи для вас, щоб залучити ваших користувачів.

Наприклад, ваш додаток для туризму може автоматично додавати маршрути в календарі користувача. Ви можете використовувати Google Calendar API, щоб знайти і переглядати публічні події календаря. Якщо ви авторизований, ви також можете отримати доступ і змінювати особисті календарі та події на цих календарів.

Використовуйте Calendar API для досягнення більш глибокої інтеграції з Google Calendar. Мобільні додатки, веб-додатки та інші системи можуть створювати, переглядати, або синхронізувати з даними календаря.

Calendar API є REST API, які можуть бути доступні через явні виклики HTTP або за допомогою Google Client Libraries; API-інтерфейс надає більшість функцій, доступних в веб версії Google Calendar.

Для використання Google сервісів на стороні серверу потрібен клієнт і Google надає API для роботи з ним. Давайте ознайомимося з ним. Головна точка входу для інтеграції сервісів Google Play.

GoogleApiClient використовується з різними статичними методами. Деякі з методів вимагають, щоб GoogleApiClient був підключеним, деякі з них будуть чекати в черзі викликів, перш ніж GoogleApiClient буде підключений [27].

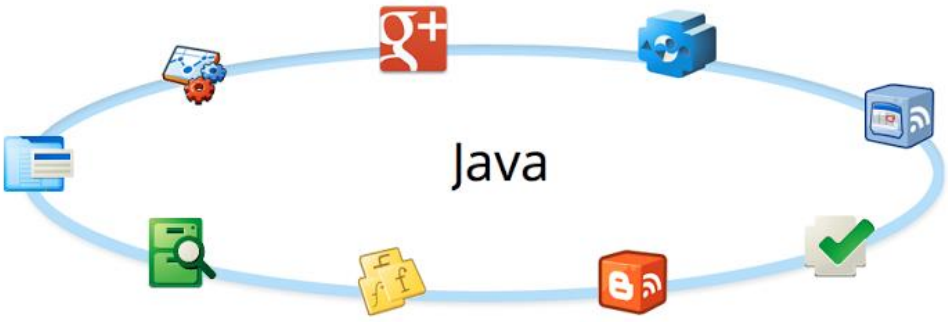


Рис.4.4 Зручний доступ до Google API за допомогю Java [26]

Бібліотека Google API-клієнт для Java надає функціональні можливості, загальні для всіх Google API, наприклад, HTTP транспорту, обробки помилок,

автентифікації, JSON синтаксичного аналізу, медіа-завантаження. Бібліотека включає в себе потужну бібліотеку OAuth 2.0; легкі, ефективні моделі даних XML і JSON, які підтримують будь-яку схему даних; і підтримка для буферів протоколу.

Для виклику Google API, використовуючи клієнтські бібліотеки Google, для Java необхідно мати згенеровану бібліотеку Java для Google API з якої ви звертаєтеся. Такі бібліотеки включають в себе основу google-api-java-client бібліотеки разом з інформацією API-специфічною, таку як коренева URL. Вони також включають в себе класи в контексті API, що можуть бути використані для створення переходів між об'єктами JSON і об'єктами Java.

Ось приклад, який використовує Calendar API Client Library для Java, щоб зробити виклик Google Calendar API:

```
// Show events on user's calendar.
View.header("Show Calendars");
CalendarList feed = client.calendarList().list().execute();
View.display(feed);
```

Бібліотека включає в себе потужну бібліотеку автентифікації, яка може зменшити обсяг коду, який потрібно обробляти OAuth 2.0. Іноді кілька рядків все, що вам потрібно. Наприклад:

```
 /** Authorizes the installed application to access user's protected data.
*/
 private static Credential authorize() throws Exception {
   // load client secrets
   GoogleClientSecrets clientSecrets =
GoogleClientSecrets.load(JSON_FACTORY,
       new
InputStreamReader(CalendarSample.class.getResourceAsStream("/client_secrets.
json")));
   // set up authorization code flow
   GoogleAuthorizationCodeFlow flow = new
GoogleAuthorizationCodeFlow.Builder(
       httpTransport, JSON_FACTORY, clientSecrets,
       Collections.singleton(CalendarScopes.CALENDAR)).setDataStoreFactory(d
ataStoreFactory)
      .build();
   // authorize
   return new AuthorizationCodeInstalledApp(flow, new
```

```
LocalServerReceiver()).authorize("user");
}
```

4.7 Інтерфейс користувача

В даному розділі розглянемо основні елементи інтерфейсу розробленого прототипу, які дозволяють користувачу ефективно вирішувати задачу створення індивідуального календарного плану навчального процесу. Першим етапом роботи є автентифікація в системі, що відбувається в окремому діалоговому вікні. Користувач повинен ввести власні логін та пароль для надання доступу для роботи в інформаційній системі.

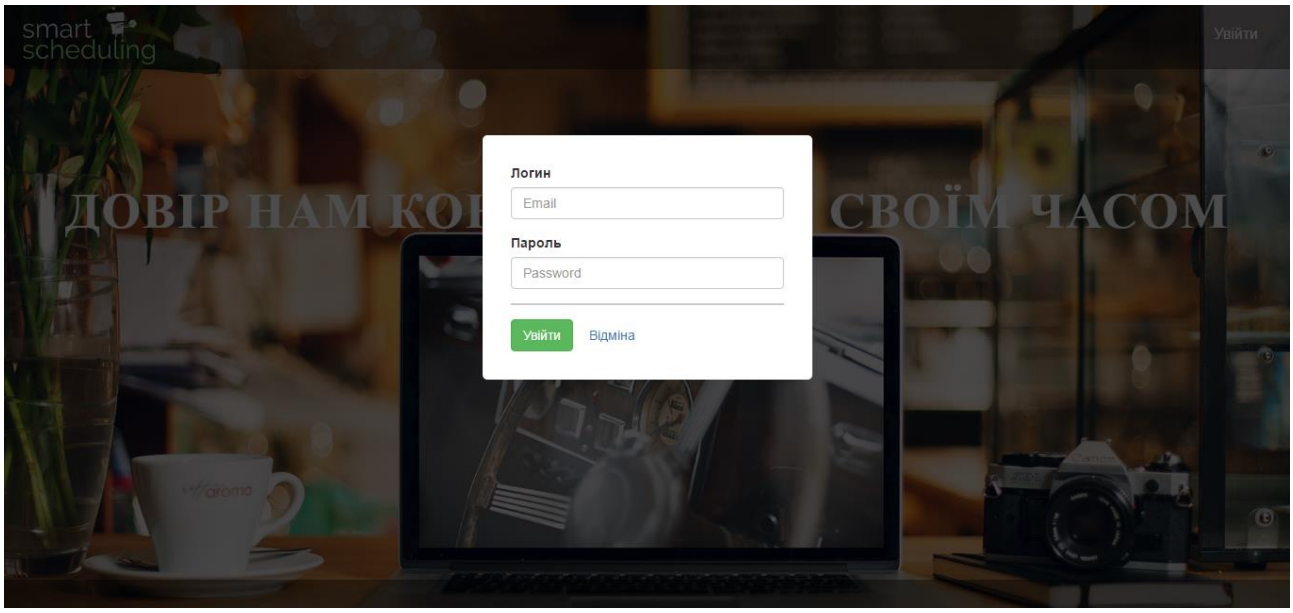


Рис.4.5 Авторизація користувача в системі

Після процесу автентифікації користувач потрапляє у власний кабінет, де він має змогу переглядати задачі навчального плану.

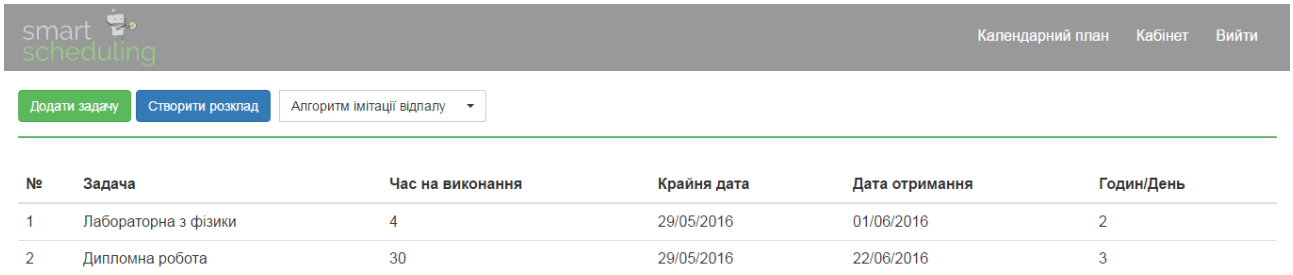


Рис. 4.6 Кабінет користувача. Перегляд задач навчального плану.

За допомогою окремого діалогового вікна у користувача є можливість додавати задачі до свого плану. Для цього необхідно вказати наступні дані про задачу:

- Назва задачі.
- Крайній строк здачі.
- Час на виконання даної задачі.
- Бажана кількість годин, яка буде витрачатись на дану задачу.
- Пріоритет задачі.

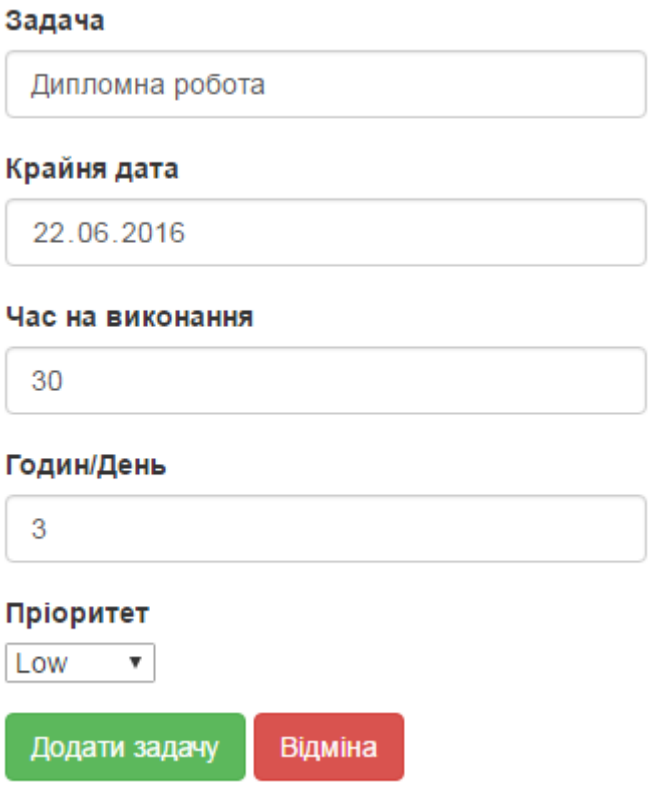


Рис.4.7 Діалогове вікно для додавання задач до календарного плану

Також в кабінеті є можливість обирати алгоритм оптимізації календарного плану.

Після натискання на кнопку “Створити розклад” користувач автоматично потрапляє на сторінку зі сформованим календарним планом, де він має можливість переглянути які задачі йому необхідно виконувати сьогодні та на який день запланована та чи інша активність. Фінальний розклад представлений у вигляді календаря з можливістю переглядати різні представлення (увесь місяць, тиждень, окремий день). Синім кольором в полі

дня позначені задачі, які повинні бути виконаними в даний день. Також є змога редагувати готовий календарний план за власним бажанням (перетягувати задачі з одного дня на інший, змінювати час початку та кінця виконання задачі, змінювати тривалість задачі).

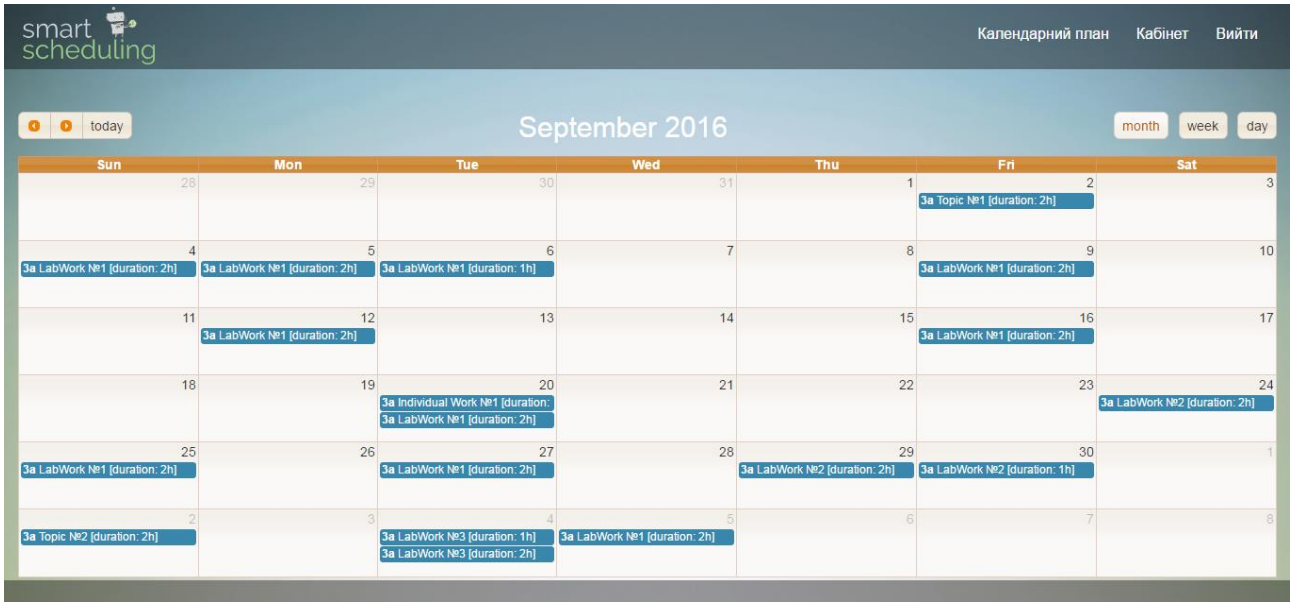


Рис.4.8 Оптимізований індивідуальний календарний план

4.7 Висновки

В даному розділі розглянуті основні переваги обраних технологій для реалізації прототипу системи та їх порівняльна характеристика з іншими альтернативами. Для вирішення задачі довгострокового збереження даних була обрана БД PostgreSQL. Взаємодія клієнта та сервера реалізована з використанням Spring MVC та Spring REST для реалізації API. Для зменшення зв’язаності модулів та спрощення тестування прототипу використовувався Spring IOC контейнер. Для реалізації шару сервісів, що відповідають за доступ до даних використовувався фреймворк Spring Data. Контроль якості розроблюваного додатку забезпечувався юніт тестуванням з використанням фреймворку JUnit.

5. ПОРІВНЯЛЬНА ХАРАКТЕРИСТИКА РЕЗУЛЬТАТІВ РОБОТИ АЛГОРИТМІВ КАЛЕНДАРНОГО ПЛАНУВАННЯ

Насамперед розглянемо функцію температури для алгоритму імітації відпалу. З попередніх розділів відомо, що функція температури визначає кількість ітерацій роботи алгоритму та зі зменшенням значення температури зменшується ймовірність прийняття гіршого рішення, тобто зменшується кількість степенів свободи в процесі прийняття рішення. Також експериментальним шляхом була встановлено, що початковий “прогрів” приводить алгоритм до кращого фінального результату. Під поняттям прогріву мається на увазі стала початкова температура протягом перших ітерацій (в даному випадку перших 300 ітерацій зберігається стала температура 10 000 градусів). Початкова температура відповідає позначці 10000 градусів.

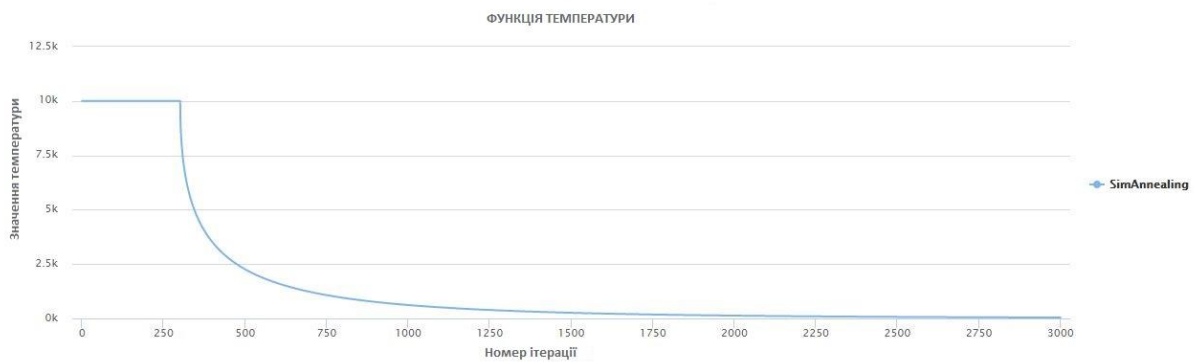


Рис.5.1 Функція температури для алгоритму імітації відпалу

Розглянемо також залежність ймовірності прийняття рішення на кожній з ітерацій. Якщо наступний згенерований розклад має кращу функцію енергії (тобто її значення менше від попереднього), тоді ймовірність прийняття рішення 100% (тобто краще рішення завжди буде обраним). Ймовірнісна модель алгоритму полягає в тому, що у випадку коли на наступній ітерації маємо розклад з більшим значенням фітнес функції – необхідно визначити з якою ймовірністю приймати гірше рішення. З ростом номеру ітерації зменшується значення температури, а отже і зменшується ймовірність

прийняття гіршого рішення. Дана поведінка алгоритму відображена в нижченаведених графіках.

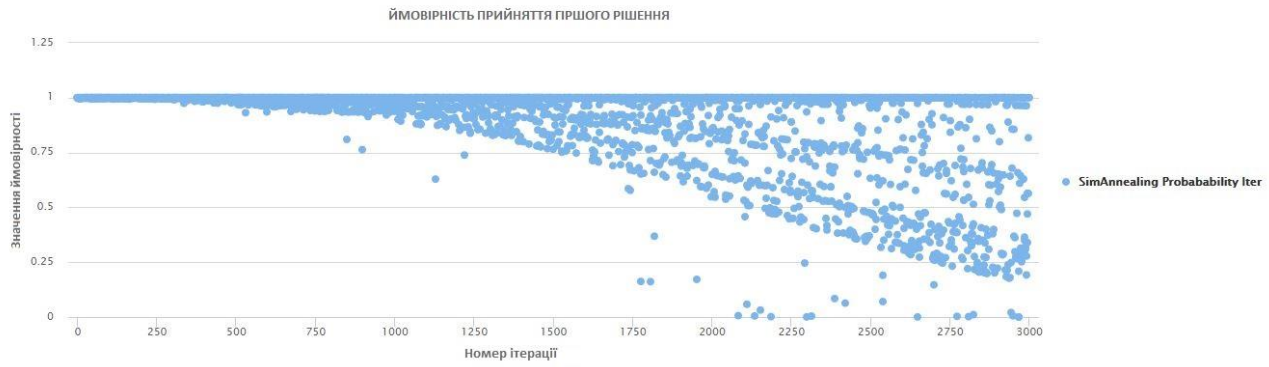


Рис.5.2 Залежність ймовірності прийняття рішення від номеру ітерації

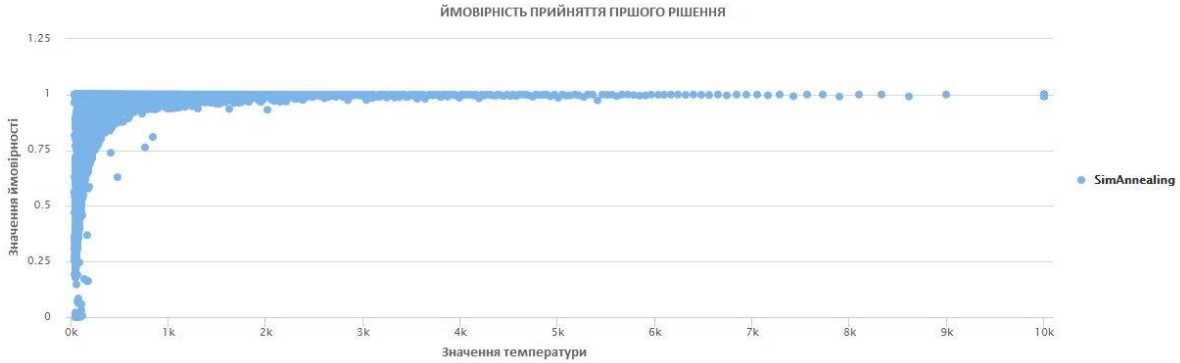


Рис.5.3 Залежність ймовірності прийняття рішення від значення температури

Порівняємо штрафну функцію для алгоритму імітації відпалу та жадібного алгоритму.

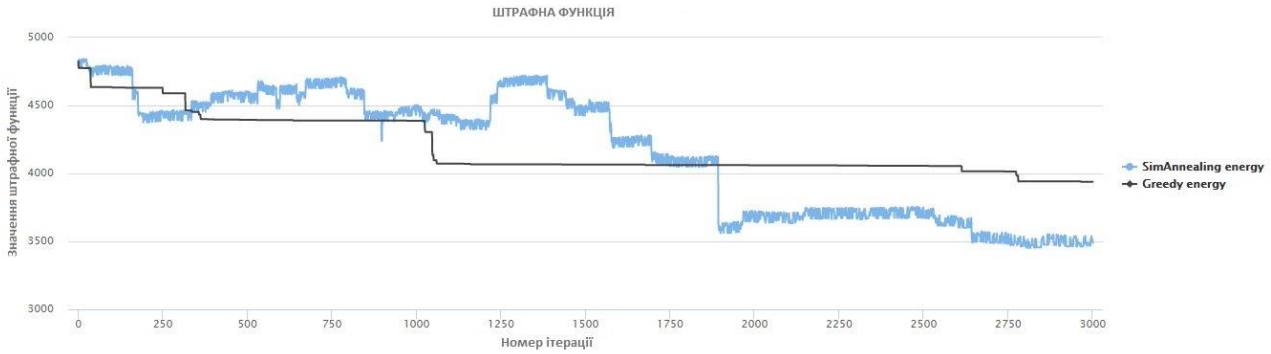


Рис.5.4 Штрафна функція для алгоритму імітації відпалу та жадібного алгоритму

Значення функції енергії/штрафної функції для початкового розкладу 4825.098. З графіку видно, що жадібний алгоритм приймає лише локально оптимальні рішення, проте в результаті такої поведінки може залишатися в рамках локального мінімуму без змоги вийти за його межі. Бачимо що з 1063 по 2606 ітерацію жадібний алгоритм перебував в рамках локального мінімуму та не мав змоги вийти за його межі. Натомість алгоритм імітації відпалу при високих значеннях температури часто приймав локально неоптимальні рішення, проте в результаті така поведінка дала змогу просунутися далі в пошуках глобального мінімуму.

Жадібний алгоритм в результаті своєї роботи мінімізував значення штрафної функції з позначки 4825.098 до 3939.222, що складає

$$\frac{(4825.098 - 3839.222)}{4825.098} * 100\% = 20.4\%$$

Тобто початковий розклад було оптимізовано на 20.4%.

Алгоритм імітації відпалу в результаті своєї роботи мінімізував значення функції енергії з позначки 4825.098 до 3387.516, що складає

$$\frac{(4825.098 - 3387.516)}{4825.098} * 100\% = 30\%$$

Тобто початковий розклад було оптимізовано на 30%.

5.1 Порівняння з результатами роботи генетичного алгоритму

Розглянемо також результати ще одного тестування, проте вже в порівнянні з результатами роботи генетичного алгоритму.

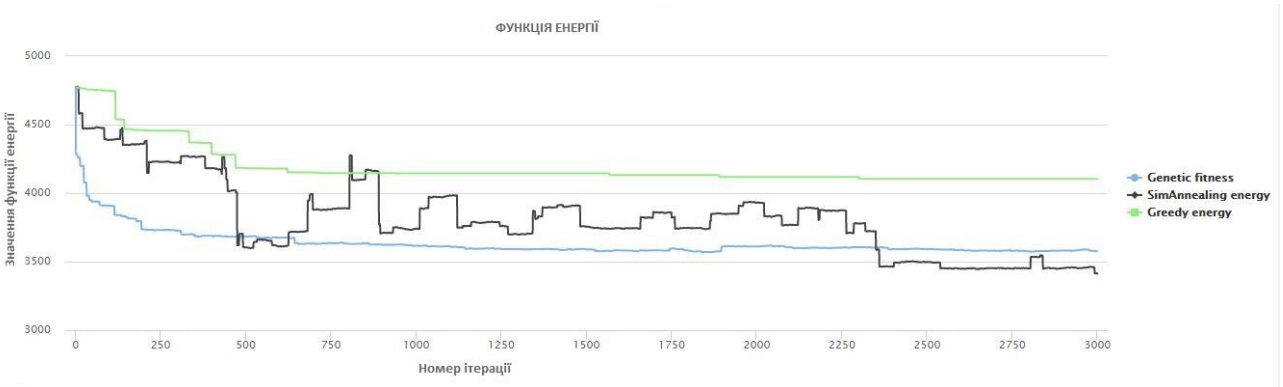


Рис.5.5 Порівняння з результатами роботи генетичного алгоритму

Жадібний алгоритм в результаті своєї роботи мінімізував значення штрафної функції з позначки 4795.110 до 4004.337, що складає

$$\frac{(4795.110 - 4004.337)}{4795.110} * 100\% = 17\%$$

Алгоритм імітації відпалу в результаті своєї роботи мінімізував значення функції енергії з позначки 4825.098 до 3387.516, що складає

$$\frac{(4795.110 - 3412.758)}{4795.110} * 100\% = 29\%$$

Генетичний алгоритм в результаті своєї роботи мінімізував значення функції енергії з позначки 4825.098 до 3387.516, що складає

$$\frac{(4795.110 - 3570.550)}{4795.110} * 100\% = 26\%$$

5.2 Висновки

В даному розділі наведено порівняльну характеристику роботи алгоритмів у вигляді графіків. Для методу імітації відпалу було відображено процес розігріву та швидкість охолодження системи, продемонстрована залежність ймовірності прийняття гіршого рішення на кожній ітерації від значення температури. Зі зниженням температури підвищується інертність системи і ймовірність прийняття гіршого рішення різко знижується. Графік зі значеннями штрафної функції демонструє проблему жадібного вибору, в результаті якого можливе потрапляння в локальний мінімум без можливості продовження пошуку глобального мінімуму.

Алгоритм імітації відпалу демонструє кращі показники мінімізації штрафної функції (до 30%) на відміну від жадібного (до 20%). Це пояснюється тим, що стохастична модель прийняття рішень дозволяє з певною ймовірністю виходити з локальних мінімумів та продовжувати пошук. Алгоритм імітації відпалу має досить велику кількість важелів конфігурації, тобто надає можливість налаштувати його параметри під конкретну задачу.

Також в даному розділі наведена порівняльна характеристика з генетичним алгоритмом. Його показники мінімізації (до 26%) близькі до алгоритму імітації відпалу. Проте генерація великої кількість генів в популяції призводить до використання пам’яті розміром пропорційним кількості генів в усіх популяціях, що були задіяні. Це призводить до частих запусків збирача сміття, в результаті чого можливі просідання в швидкості видачі готового результату кінцевому користувачу на великих періодах планування.

6. ФУНКЦІОНАЛЬНО-ВАРТІСНИЙ АНАЛІЗ ПРОГРАМНОГО ПРОДУКТУ

У даному розділі проводиться оцінка основних характеристик програмного продукту, Smart Scheduling. Інтерфейс користувача був розроблений за допомогою мови програмування Java у середовищі розробки Intellij IDEA. Інтерфейс користувача створений за допомогою технології Spring.

Програмний продукт призначено для використання на персональних комп'ютерах під управлінням будь-якої операційної системи.

Нижче наведено аналіз різних варіантів реалізації модулю з метою вибору оптимальної, з огляду при цьому як на економічні фактори, так і на характеристики продукту, що впливають на продуктивність роботи і на його сумісність з апаратним забезпеченням. Для цього було використано апарат функціонально-вартісного аналізу.

Функціонально-вартісний аналіз (ФВА) – це технологія, яка дозволяє оцінити реальну вартість продукту або послуги незалежно від організаційної структури компанії. Як прямі, так і побічні витрати розподіляються по продуктам та послугам у залежності від потрібних на кожному етапі виробництва обсягів ресурсів. Виконані на цих етапах дії у контексті метода ФВА називаються функціями.

Мета ФВА полягає у забезпеченні правильного розподілу ресурсів, виділених на виробництво продукції або надання послуг, на прямі та непрямі витрати. У даному випадку – аналізу функцій програмного продукту й виявлення усіх витрат на реалізацію цих функцій.

Фактично цей метод працює за таким алгоритмом:

– визначається послідовність функцій, необхідних для виробництва продукту. Спочатку – всі можливі, потім вони розподіляються по двом групам: ті, що впливають на вартість продукту і ті, що не впливають. На цьому ж етапі оптимізується сама послідовність скороченням кроків, що не впливають на цінність і відповідно витрат.

– для кожної функції визначаються повні річні витрати й кількість робочих часів.

– для кожної функції наоснові оцінок попереднього пункту визначається кількісна характеристика джерел витрат.

– після того, як для кожної функції будуть визначені їх джерела витрат, проводиться кінцевий розрахунок витрат на виробництво продукту.

6.1 Постановка задачі техніко-економічного аналізу

У роботі застосовується метод ФВА для проведення техніко-економічний аналізу розробки.

Відповідно цьому варто обирати і систему показників якості програмного продукту.

Технічні вимоги до продукту наступні:

– програмний продукт повинен функціонувати на персональних комп’ютерах із стандартним набором компонент;

– забезпечувати високу швидкість обробки великих об’ємів даних у реальному часі;

– забезпечувати зручність і простоту взаємодії з користувачем або з розробником програмного забезпечення у випадку використовування його як модуля;

– передбачати мінімальні витрати на впровадження програмного продукту.

6.1.1 Обґрунтування функцій програмного продукту

Головна функція F0– розробка програмного продукту, який аналізує процес за вхідними даними та будує його модель для подальшого прогнозування. Виходячи з конкретної мети, можна виділити наступні основні функції ПП:

F1 – вибір мови програмування;

F2 – вибір оптимальної СКБД;

F3 – інтерфейс користувача.

Кожна з основних функцій може мати декілька варіантів реалізації.

Функція F1:

а) мова програмування C#;

б) мова програмування Java;

Функція F2:

а) MySQL;

б) PostgreSQL.

Функція F3:

а) інтерфейс користувача, створений за технологією ASP Net;

б) інтерфейс користувача, створений за технологією Spring.

6.1.2 Варіанти реалізації основних функцій

Варіанти реалізації основних функцій наведені у морфологічній карті системи (рис. 4.1). На основі цієї карти побудовано позитивно-негативну матрицю варіантів основних функцій (таблиця 4.1).

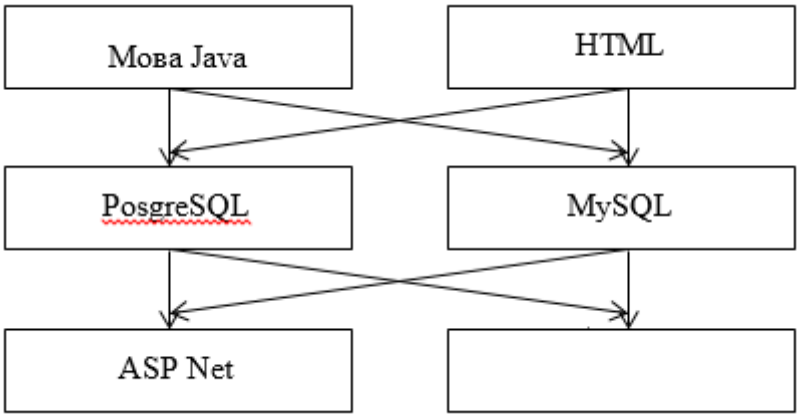


Рисунок 6.1 – Морфологічна карта

Морфологічна карта відображує всі можливі комбінації варіантів реалізації функцій, які складають повну множину варіантів ПП.

Таблиця 6.1 – Позитивно-негативна матриця

Основні функції	**Варіанти реалізації**	**Переваги**	**Недоліки**
F1	*А*	Кросплатформений	Низька швидкодія
	Б	Займає менше часу при написанні коду	Більший час на виконання операцій
F2	А	Безкоштовність	Відсутність вкладених запитів
	Б	Надійність, внесення змін без перезапуску, безкоштовність	Необхідність додаткової інсталяції, низький рівень користувацької підтримки
F3	*А*	Простота створення	Відсутність кросплатформеності
	Б	Простота створення,	Кросплатформеність

На основі аналізу позитивно-негативної матриці робимо висновок, що при розробці програмного продукту деякі варіанти реалізації функцій варто відкинути, тому, що вони не відповідають поставленим перед програмним продуктом задачам. Ці варіанти відзначені у морфологічній карті.

Функція *F1:*

Оскільки розрахунки проводяться з великими об’ємами вхідних даних, то час виконання програмного коду є дуже необхідним, тому варіант а) має бути відкинутий.

Функція *F2:*

Вибір СКБД не відіграє велику роль у даному програмному продукту, тому вважаємо варіанти а) та б) гідними розгляду.

Функція *F3:*

Оскільки, програмний продукт реалізується на мові Java, використовуємо варіант Б як єдиний можливий.

Таким чином, будемо розглядати такі варіанти реалізації ПП:

1) F1б – F2а – F3б
2) F1б – F2б – F3б

Для оцінювання якості розглянутих функцій обрана система параметрів, описана нижче.

6.2 Обґрунтування системи параметрів ПП

6.2.1 Опис параметрів

На підставі даних про основні функції, що повинен реалізувати програмний продукт, вимог до нього, визначаються основні параметри виробу, що будуть використані для розрахунку коефіцієнта технічного рівня.

Для того, щоб охарактеризувати програмний продукт, будемо використовувати наступні параметри:

– *X1* – швидкодія мови програмування;

– *X2* – об’єм пам’яті для збереження даних;

– *X3* – час обробки даних;

– *X4* – потенційний об’єм програмного коду.

X1: Відображає швидкодію операцій залежно від обраної мови програмування.

X2: Відображає об’єм пам’яті в оперативній пам’яті персонального комп’ютера, необхідний для збереження та обробки даних під час виконання програми.

X3: Відображає час, який витрачається на дії.

X4: Показує розмір програмного коду який необхідно створити безпосередньо розробнику.

6.2.2 Кількісна оцінка параметрів

Гірші, середні і кращі значення параметрів вибираються на основі вимог замовника й умов, що характеризують експлуатацію ПП як показано у табл. 4.2.

Таблиця 6.2 – Основні параметри ПП

Назва Параметра	**Умовні позначення**	**Одиниці виміру**	**Значення параметра**		
			гірші	**середні**	**кращі**
Швидкодія мови програмування	X1	Оп/мс	19000	11000	2000
Об’єм пам’яті для збереження даних	X2	Мб	32	16	8
Час обробки запитів користувача	X3	мс	1000	420	60
Потенційний об’єм програмного коду	X4	кількість строк коду	2000	1500	1000

За даними таблиці 6.2 будуються графічні характеристики параметрів – рис. 6.2 – рис. 6.5.

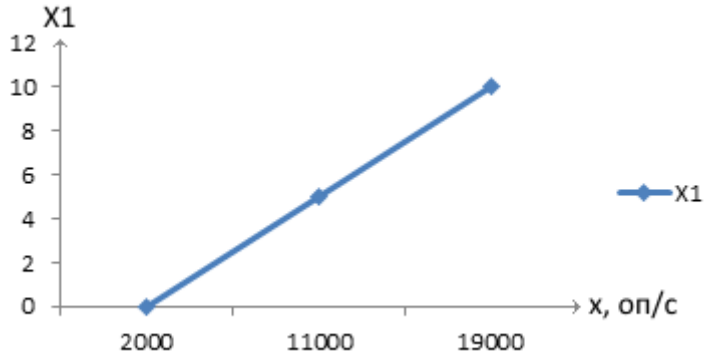


Рисунок 6.2 – X1, швидкодія мови програмування

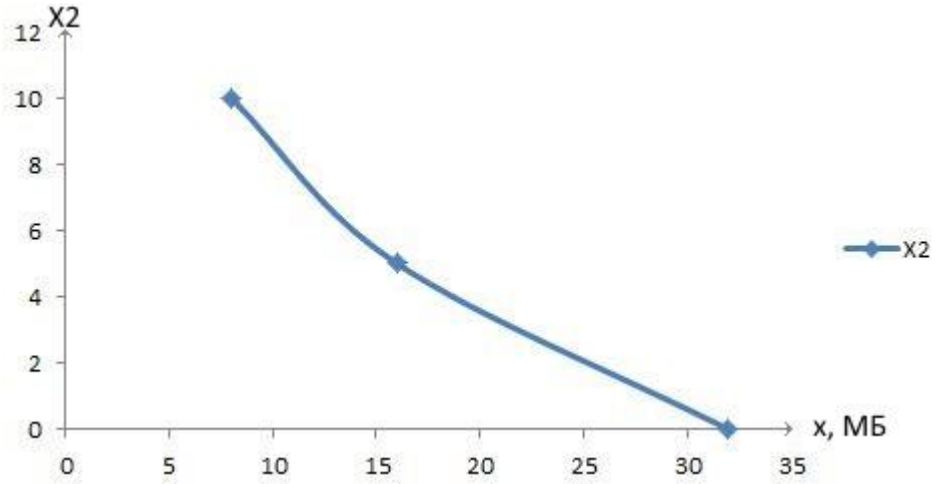


Рисунок 6.3 – X2, об’єм пам’яті для збереження даних

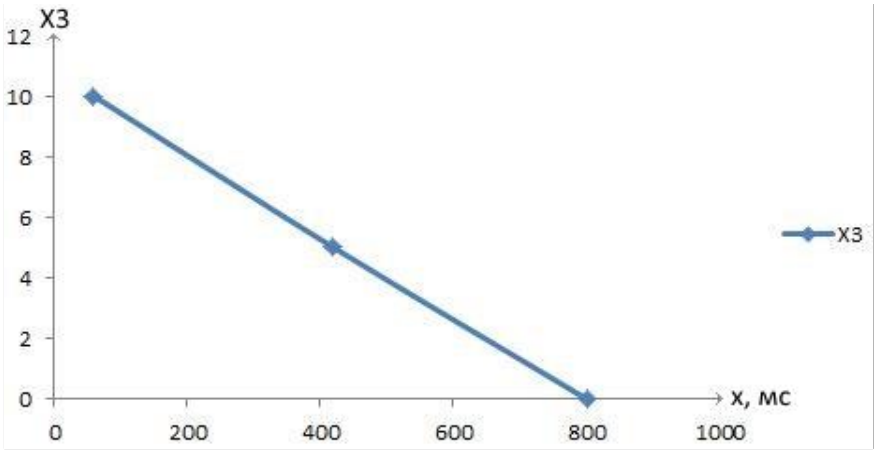


Рисунок 6.4 – X3, час виконання запитів користувача

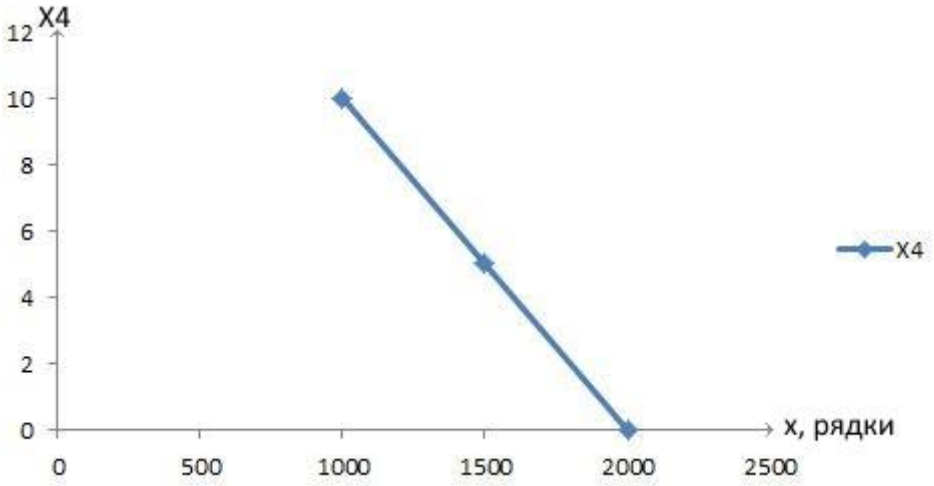


Рисунок 6.5 – X4, потенційний об'єм програмного коду

6.2.3 Аналіз експертного оцінювання параметрів

Після детального обговорення й аналізу кожний експерт оцінює ступінь важливості кожного параметру для конкретно поставленої цілі – розробка програмного продукту, який дає найбільш точні результати при знаходженні параметрів моделей адаптивного прогнозування і обчислення прогнозних значень.

Значимість кожного параметра визначається методом попарного порівняння. Оцінку проводить експертна комісія із 7 людей. Визначення коефіцієнтів значимості передбачає:

– визначення рівня значимості параметра шляхом присвоєння різних рангів;

– перевірку придатності експертних оцінок для подальшого використання;

– визначення оцінки попарного пріоритету параметрів;

– обробку результатів та визначення коефіцієнту значимості.

Результати експертного ранжування наведені у таблиці 4.3.

Таблиця 6.3 – Результати ранжування параметрів

Позначення параметра	Назва параметра	Одиниці виміру	Ранг параметра за оцінкою експерта							Сума рангів R_i	Відхилення Δ_i	Δ_i^2
			1	2	3	4	5	6	7			
X1	Швидкодія мови програмування	Оп/мс	4	3	4	4	4	4	4	27	0,75	0,56
X2	Об’єм пам’яті для збереження даних	Мб	4	4	4	3	4	3	3	25	-1,25	1,56
X3	Час обробки запитів користувача	Мс	2	2	1	2	1	2	2	12	-14,25	203,06
X4	Потенційний об’єм програмного коду	кількість строк коду	5	6	6	6	6	6	6	41	14,75	217,56
	Разом		15	15	15	15	15	15	15	105	0	420,75

Для перевірки степені достовірності експертних оцінок, визначимо наступні параметри:

а) сума рангів кожного з параметрів і загальна сума рангів:

$$R_i = \sum_{j=1}^{N} r_{ij} R_{ij} = \frac{Nn(n+1)}{2} = 105,$$

де N – число експертів, n – кількість параметрів;

б) середня сума рангів:

$$T = \frac{1}{n} R_{ij} = 26{,}25.$$

в) відхилення суми рангів кожного параметра від середньої суми рангів:

$$\Delta_i = R_i - T$$

Сума відхилень по всім параметрам повинна дорівнювати 0;

г)загальна сума квадратів відхилення:

$$S = \sum_{i=1}^{N} \Delta_i^2 = 420{,}75.$$

Порахуємо коефіцієнт узгодженості:

$$W = \frac{12S}{N^2(n^3 - n)} = \frac{12 \cdot 420{,}75}{7^2(5^3 - 5)} = 1{,}03 > W_k = 0{,}67$$

Ранжування можна вважати достовірним, тому що знайдений коефіцієнт узгодженості перевищує нормативний, котрий дорівнює 0,67.

Скориставшись результатами ранжирування, проведемо попарне порівняння всіх параметрів і результати занесемо у таблицю4.4.

Таблиця 6.4 – Попарне порівняння параметрів

Параметри	Експерти							Кінцева оцінка	Числове значення
	1	2	3	4	5	6	7		
X1 і X2	=	>	=	<	=	<	<	<	0,5
X1 і X3	<	<	<	<	<	<	<	<	0,5
X1 і X4	>	>	>	>	>	>	>	>	1,5
X2 і X3	<	<	<	<	<	<	<	<	0,5
X2 і X4	>	>	>	>	>	>	>	>	1,5
X3 і X4	>	>	>	>	>	>	>	>	1,5

Числове значення, що визначає ступінь переваги *i*–го параметра над *j*–тим, a_{ij} визначається по формулі:

$$a_{ij} = \begin{cases} 1{,}5 \text{ при } X_i > X_j \\ 1.0 \text{ при } X_i = X_j \\ 0.5 \text{ при } X_i < X_j \end{cases}$$

З отриманих числових оцінок переваги складемо матрицю $A = \| a_{ij} \|$.

Для кожного параметра зробимо розрахунок вагомості $K_{ві}$ за наступними формулами:

$K_{\text{ві}} = \frac{b_i}{\sum_{i=1}^{n} b_i}$,де $b_i = \sum_{i=1}^{N} a_{ij}$.

Відносні оцінки розраховуються декілька разів доти, поки наступні значення не будуть незначно відрізнятися від попередніх (менше 2%).На другому і наступних кроках відносні оцінки розраховуються за наступними формулами:

$K_{\text{ві}} = \frac{b_i'}{\sum_{i=1}^{n} b_i'}$,де $b_i' = \sum_{i=1}^{N} a_{ij} b_j$.

Як видно з таблиці 4.5, різниця значень коефіцієнтів вагомості не перевищує 2%, тому більшої кількості ітерацій не потрібно.

Таблиця 6.5 – Розрахунок вагомості параметрів

Параметриx_i	Параметриx_j				Перша ітер.		Друга ітер.		Третя ітер	
	X1	X2	X3	X4	b_i	$K_{\text{ві}}$	b_i^1	$K_{\text{ві}}^1$	b_i^2	$K_{\text{ві}}^2$
X1	1,0	0,5	0,5	1,5	3,5	0,219	22,25	0,216	100	0,215
X2	1,5	1,0	0,5	1,5	4,5	0,281	27,25	0,282	124,25	0,283
X3	1,5	1,5	1,0	1,5	5,5	0,344	34,25	0,347	156	0,348
X4	0,5	0,5	0,5	1,0	2,5	0,156	14,25	0,155	64,75	0,154
Всього:					16	1	98	1	445	1

6.3 Аналіз рівня якості варіантів реалізації функцій

Визначаємо рівень якості кожного варіанту виконання основних функцій окремо.

Абсолютні значення параметрів *X2*(об’єм пам’яті для збереження даних) та *X1* (швидкодія мови програмування) відповідають технічним вимогам умов функціонування даного ПП.

Абсолютне значення параметра *X3 (*час обробки даних) обрано не найгіршим (не максимальним), тобто це значення відповідає або варіанту а) 1000 мс або варіанту б) 80мс.

Коефіцієнт технічного рівня для кожного варіанта реалізації ПП розраховується так (таблиця 4.6):

$$K_K(j) = \sum_{i=1}^{n} K_{вi,j} B_{i,j} ,$$

де n – кількість параметрів;$K_{вi}$– коефіцієнт вагомості i–го параметра;B_i – оцінка i–го параметра в балах.

Таблиця 6.6 – Розрахунок показників рівня якості варіантів реалізації основних функцій ПП

Основні функції	Варіант реалізації функції	Абсолютне значення параметра	Бальна оцінка параметра	Коефіцієнт вагомості параметра	Коефіцієнт рівня якості
F1(X1)	А	11000	3,6	0,215	0,774
F2(X2)	А	16	3,4	0,283	0,962
F3(X3,X4)	А	800	2,4	0,348	0,835
	Б	80	1	0,154	0,154

За даними з таблиці 4.6 за формулою

$$K_K = K_{\text{ТУ}}[F_{1k}] + K_{\text{ТУ}}[F_{2k}] + \ldots + K_{\text{ТУ}}[F_{zk}],$$

визначаємо рівень якості кожного з варіантів:

K_{K1} = 0,774 + 0,962 + 0,835 = 2,57

K_{K2} = 0,774 + 0,962 + 0,154 = 1,89

Як видно з розрахунків, кращим є перший варіант, для якого коефіцієнт технічного рівня має найбільше значення.

6.4 Економічний аналіз варіантів розробки ПП

Для визначення вартості розробки ПП спочатку проведемо розрахунок трудомісткості.

Всі варіанти включають в себе два окремих завдання:

1. Розробка проекту програмного продукту;

2. Розробка програмної оболонки;

Завдання 1 за ступенем новизни відноситься до групи А, завдання 2 – до групи Б. За складністю алгоритми, які використовуються в завданні 1 належать до групи 1; а в завданні 2 – до групи 3.

Для реалізації завдання 1 використовується довідкова інформація, а завдання 2 використовує інформацію у вигляді даних.

Проведемо розрахунок норм часу на розробку та програмування для кожного з завдань.

Проведемо розрахунок норм часу на розробку та програмування для кожного з завдань. Загальна трудомісткість обчислюється як

$$T_O = T_P \cdot K_П \cdot K_{СК} \cdot K_М \cdot K_{СТ} \cdot K_{СТ.М}, \quad (5.1)$$

де T_P – трудомісткість розробки ПП;$K_П$ – поправочний коефіцієнт;$K_{СК}$ – коефіцієнт на складність вхідної інформації; $K_М$ – коефіцієнт рівня мови програмування;$K_{СТ}$ – коефіцієнт використання стандартних модулів і прикладних програм; $K_{СТ.М}$ – коефіцієнт стандартного математичного забезпечення.

Для першого завдання, виходячи із норм часу для завдань розрахункового характеру степеню новизни А та групи складності алгоритму 1, трудомісткість дорівнює: T_P =90 людино-днів. Поправочний коефіцієнт, який враховує вид нормативно-довідкової інформації для першого завдання: $K_П$ = 1.7. Поправочний коефіцієнт, який враховує складність контролю вхідної та вихідної інформації для всіх семи завдань рівний 1: $K_{СК}$ = 1. Оскільки при розробці першого завдання використовуються стандартні модулі, врахуємо це за допомогою коефіцієнта $K_{СТ}$ = 0.8. Тоді, за формулою 5.1, загальна трудомісткість програмування першого завдання дорівнює:

$T_1 = 90 \cdot 1.7 \cdot 0.8 = 122.4$ людино-днів.

Проведемо аналогічні розрахунки для подальших завдань.

Для другого завдання (використовується алгоритм третьої групи складності, степінь новизни Б), тобто T_P =27 людино-днів, $K_П$ =0.9,$K_{СК}$ = 1,$K_{СТ}$ =0.8:

$T_2 = 27 \cdot 0.9 \cdot 0.8 = 19.44$ людино-днів.

Складаємо трудомісткість відповідних завдань для кожного з обраних варіантів реалізації програми, щоб отримати їх трудомісткість:

$T_I = (122.4 + 19.44 + 4.8 + 19.44) \cdot 8 = 1328{,}64$ людино-годин;

$T_{II} = (122.4 + 19.44 + 6.91 + 19.44) \cdot 8 = 1345.52$ людино-годин;

Найбільш високу трудомісткість має варіант II.

В розробці беруть участь два програмісти з окладом 7000 грн., один фінансовий аналітик з окладом 9500грн. Визначимо зарплату за годину за формулою:

$$С_ч = \frac{М}{T_m \cdot t} грн.,$$

де М – місячний оклад працівників;T_m – кількість робочих днів тиждень;t – кількість робочих годин в день.

$$С_ч = \frac{7000 + 7000 + 9500}{3 \cdot 21 \cdot 8} = 46{,}62 \text{ грн.}$$

Тоді, розрахуємо заробітну плату за формулою

$$С_{ЗП} = С_ч \cdot Т_i \cdot К_Д,$$

де $С_ч$– величина погодинної оплати праці програміста;$Т_i$ – трудомісткість відповідного завдання; $К_Д$ – норматив, який враховує додаткову заробітну плату.

Зарплата розробників за варіантами становить:

I. $С_{ЗП}$ = 46,62· 1328.64 · 1.2 = 74340,57 грн.

II. $С_{ЗП}$ = 46,62· 1345.52 · 1.2 = 75285,04 грн.

Відрахування на єдиний соціальний внесок в залежності від групи професійного ризику (II клас) становить 22%:

I. $С_{ВІД} = С_{ЗП} \cdot 0.22$ = 74340,57*0.22 = 16354.93 грн.

II. $С_{ВІД} = С_{ЗП} \cdot 0.22$ = 75285,04*0.22 =16562.71 грн.

Тепер визначимо витрати на оплату однієї машино-години. ($С_М$)

Так як одна ЕОМ обслуговує одного програміста з окладом 7000 грн., з коефіцієнтом зайнятості 0,2 то для однієї машини отримаємо:

$С_Г = 12 \cdot М \cdot К_З$ = 12 · 7000· 0,2 = 16800 грн.

З урахуванням додаткової заробітної плати:

$С_{ЗП} = С_Г \cdot (1 + К_З)$ = 16800· (1 + 0.2) = 20160 грн.

Відрахування на єдиний соціальний внесок:

$С_{ВІД} = С_{ЗП} \cdot 0.22$= 20160 * 0,22 = 4435.20 грн.

Амортизаційні відрахування розраховуємо при амортизації 25% та вартості ЕОМ – 10000 грн.

$С_А = К_{ТМ} \cdot К_А \cdot Ц_{ПР} = 1.15 \cdot 0.25 \cdot 10000 = 2875$ грн.,

де $К_{ТМ}$– коефіцієнт, який враховує витрати на транспортування та монтаж приладу у користувача; $К_А$– річна норма амортизації; $Ц_{ПР}$– договірна ціна приладу.

Витрати на ремонт та профілактику розраховуємо як:

$С_Р = К_{ТМ} \cdot Ц_{ПР} \cdot К_Р = 1.15 \cdot 10000 \cdot 0.05 = 575$ грн.,

де $К_Р$– відсоток витрат на поточні ремонти.

Ефективний годинний фонд часу ПК за рік розраховуємо за формулою:

$Т_{ЕФ} = (Д_К – Д_В – Д_С – Д_Р) \cdot t_З \cdot К_В = (365 – 104 – 8 – 16) \cdot 8 \cdot 0.9 = 1706.4$ годин,

де $Д_К$ – календарна кількість днів у році; $Д_В$, $Д_С$ – відповідно кількість вихідних та святкових днів; $Д_Р$ – кількість днів планових ремонтів устаткування; t –кількість робочих годин в день; $К_В$– коефіцієнт використання приладу у часі протягом зміни.

Витрати на оплату електроенергії розраховуємо за формулою:

$С_{ЕЛ} = Т_{ЕФ} \cdot N_С \cdot К_З \cdot Ц_{ЕН} = 1706{,}4 \cdot 0{,}156 \cdot 0{,}9733 \cdot 2{,}0218 = 523.83$ грн.,

де $N_С$ – середньо-споживча потужність приладу; $К_З$– коефіцієнтом зайнятості приладу; $Ц_{ЕН}$ – тариф за 1 КВт-годин електроенергії.

Накладні витрати розраховуємо за формулою:

$С_Н = Ц_{ПР} \cdot 0.67 = 10000 \cdot 0{,}67 = 6700$ грн.

Тоді, річні експлуатаційні витрати будуть:

$$С_{ЕКС} = С_{ЗП} + С_{ВІД} + С_А + С_Р + С_{ЕЛ} + С_Н$$

$С_{ЕКС} = 20160 + 4435.20 + 2875 + 575 + 523.83 + 6700 = 35269.03$ грн.

Собівартість однієї машино-години ЕОМ дорівнюватиме:

$С_{М\text{-}Г} = С_{ЕКС} / Т_{ЕФ} = 35269.03 / 1706{,}4 = 20{,}67$ грн/час.

Оскільки в даному випадку всі роботи, які пов‘язані з розробкою програмного продукту ведуться на ЕОМ, витрати на оплату машинного часу, в залежності від обраного варіанта реалізації, складає:

$$С_М = С_{М\text{-}Г} \cdot Т$$

I. $С_М = 20{,}67 * 1328{,}64 = 27462.99$ грн.;

II. $С_М = 20{,}67 * 1345.52 = 27811.9$ грн.;

Накладні витрати складають 67% від заробітної плати:

$$С_Н = С_{ЗП} \cdot 0,67$$

I. $С_Н = 74340,57 * 0,67 = 49808,18$ грн.;

II. $С_Н = 75285,04 * 0,67 = 50440,98$ грн.;

Отже, вартість розробки ПП за варіантами становить:

$$С_{ПП} = С_{ЗП} + С_{ВІД} + С_М + С_Н$$

I. $С_{ПП} = 74340,57 + 16354.93 + 27462.99 + 49808,18 = 167966.67$ грн.;

II. $С_{ПП} = 75285,04 + 16562.71 + 27811.9 + 50440,98 = 170100.63$ грн.;

6.5 Вибір кращого варіанта ПП техніко-економічного рівня

Розрахуємо коефіцієнт техніко-економічного рівня за формулою:

$$К_{ТЕРj} = К_{Кj} / С_{Фj},$$

$К_{ТЕР1} = 2,57 / 167966.67 = 0,15 \cdot 10^{-4}$;

$К_{ТЕР2} = 1,89 / 183561,43 = 0,1 \cdot 10^{-4}$;

Як бачимо, найбільш ефективним є перший варіант реалізації програми з коефіцієнтом техніко-економічного рівня $К_{ТЕР1} = 0,15 \cdot 10^{-4}$.

6.6 Висновки

В даному розділі проведено повний функціонально-вартісний аналіз ПП, який було розроблено в рамках дипломного проекту. Процес аналізу можна умовно розділити на дві частини.

В першій з них проведено дослідження ПП з технічної точки зору: було визначено основні функції ПП та сформовано множину варіантів їх реалізації; на основі обчислених значень параметрів, а також експертних оцінок їх важливості було обчислено коефіцієнт технічного рівня, який і дав змогу визначити оптимальну з технічної точки зору альтернативу реалізації функцій ПП.

Другу частину ФВА присвячено вибору із альтернативних варіантів реалізації найбільш економічно обґрунтованого. Порівняння запропонованих варіантів реалізації в рамках даної частини виконувалось за коефіцієнтом

ефективності, для обчислення якого були обчислені такі допоміжні параметри, як трудомісткість, витрати на заробітну плату, накладні витрати.

Після виконання функціонально-вартісного аналізу програмного комплексу що розроблюється, можна зробити висновок, що з альтернатив, що залишились після першого відбору двох варіантів виконання програмного комплексу оптимальним є перший варіант реалізації програмного продукту. У нього виявився найкращий показник техніко-економічного рівня якості $К_{ТЕР} = 0,15 \cdot 10^{-4}$.

Цей варіант реалізації програмного продукту має такі параметри:

– мова програмування – Java;

– СКБД PostgreSQL;

– інтерфейс користувача, створений за технологією Spring.

Даний варіант виконання програмного комплексу дає користувачу зручний інтерфейс, непоганий функціонал і швидкодію.

ВИСНОВКИ

Результатом даної роботи є аналіз особливостей та ефективності використання алгоритму імітації відпалу та жадібного алгоритму для вирішення задачі створення та оптимізації календарного плану навчального процесу. Продуктом даного аналізу є порівняльна характеристика показників роботи та результатів оптимізації розкладу на один навчальний семестр.

Були досліджені основні аналоги, які на сьогоднішній день представлені на ринку, виділені їх основні переваги та недоліки. Оскільки жодна з існуючих систем не орієнтована на врахування індивідуальних побажань користувачів при формуванні навчального плану, то було розроблено веб-орієнтований прототип такої системи та її інтеграція з сервісом Google Calendar, що використовується користувачами для управління особистим розкладом. Власне рішення включає в себе реалізацію обраних алгоритмів.

Опис предметної області та розробка основних вимог до характеристик розроблюваного прототипу системи виконаний з використанням UML діаграм прецедентів. Також проведено графічний структурний аналіз за допомогою Data Flow методології.

Наведено порівняльну характеристику роботи алгоритмів у вигляді графіків. Для методу імітації відпалу було відображено процес розігріву та швидкість охолодження системи, продемонстрована залежність ймовірності прийняття гіршого рішення на кожній ітерації від значення температури. Зі зниженням температури підвищується інертність системи і ймовірність прийняття гіршого рішення різко знижується. Графік зі значеннями штрафної функції демонструє проблему жадібного вибору, в результаті якого можливе потрапляння в локальний мінімум без можливості продовження пошуку глобального мінімуму.

Алгоритм імітації відпалу демонструє кращі показники мінімізації штрафної функції (до 30%) на відміну від жадібного (до 20%). Це пояснюється тим, що стохастична модель прийняття рішень дозволяє з певною ймовірністю виходити з локальних мінімумів та продовжувати пошук. Алгоритм імітації

відпалу має досить велику кількість важелів конфігурації, тобто надає можливість налаштувати його параметри під конкретну задачу.

Також наведена порівняльна характеристика з генетичним алгоритмом. Його показники мінімізації (до 26%) близькі до алгоритму імітації відпалу. Проте генерація великої кількість генів в популяції призводить до використання пам’яті розміром пропорційним кількості генів в усіх популяціях, що були задіяні. Це призводить до частих запусків збирача сміття, в результаті чого можливі просідання в швидкості видачі готового результату кінцевому користувачу на великих періодах планування.

ПЕРЕЛІК ПОСИЛАНЬ

1. Лазарев А.А. Решение NP-трудной задачи теории расписаний минимизации суммарного запаздывания / Лазарев А.А. // Журнал Вычислительной математики и математической физики – 2007. том 47, N.6. – С. 1087– 1098
2. Лазарев А.А. Теория расписаний. Минимизация суммарного запаздывания для одного прибора. / Лазарев А.А., Гафаров Е.Р. // Научное издание, М.: Вычислительный центр им. А.А. Дороницына РАН, 2006. - 134 с.
3. Лазарев А.А. Теория расписаний. Исследование задач с отношениями предшествования и ресурсными ограничениями. / Лазарев А.А., Гафаров Е.Р. // Научное издание, М.: Вычислительный центр им. А.А. Дороницына РАН, 2007. – 80 с.
4. Лазарев А.А. Теория расписаний задачи и алгоритмы. / Лазарев А.А., Гафаров Е.Р. - М.: Наука, 2009. – 295 с.
5. Танаев В.С. Введение в теорию расписаний. / Танаев В.С., Шкурба В.В. - М.: Наука, 1975. – 174 с.
6. Танаев В.С. Теория расписаний. Одностадийные системы. / Танаев В.С., Гордон В.С., Шафранский Я.М. - М.: Наука. Гл. ред. физ.-мат. лит., 1984. - 384 с.
7. Танаев В.С. Теория расписаний. Многостадийные системы. / Танаев В.С., Сотсков Ю.Н., Струсевич В.А. – М.: Наука. Гл. ред. физ.-мат. лит., 1989. – 328 с.
8. Танаев В.С. Теория расписаний. Групповые технологии. / Танаев В.С., Ковалев М.Я., Шафранский Я.М. - Минск: Институт технической кибернетики НАН Беларуси, 1998. - 290 с.
9. Alon N. Approximation schemes for scheduling on parallel machines / Alon N., Woeginger G.J., Yadid T. // J. of Scheduling.– 1998.– V. 1.– P. 55 – 66.

10. Van de Akker J.M. Parallel machine scheduling by column generation / Van de Akker J.M., Hoogeveen J.A., van de Velde S.L. // Oper. Res.– 1999.– V. 47, N 6.– P. 862 – 872.
11. Van de Akker J.M. Timeindexed formulations for single-machine scheduling problems: column generation / Van de Akker J.M., Hurkens C.A.J., Savelsbergh M.W.P. // INFORMS J. on Computing.– 2000.– V. 12, N 2.– P. 111 – 124.
12. Baptiste Ph. Constraint-based scheduling: applying constraint programming to scheduling problems / Baptiste Ph., Le Pape C., Nuijten W. // Kluwer Academic Publishers, 2001. – 198 p.
13. Brooks G.N. An algorithm for finding optimal or near – optimal solutions to the production scheduling problem / Brooks G.N., White C.R. // J. Ind. Eng.– 1965.– V. 16, N 1.– P. 34 – 40.
14. Brucker P. Scheduling Algorithms. / Brucker P. - Germany: Springer-Verlag 2001. - 365 p.
15. Brucker P. Complex scheduling Springer-Verlag Berlin / Brucker P., Knust S. - Heidelberg, Germany, 2006.
16. Carlier J. The one-machine sequencing problem / Carlier J. // European J. of Oper. Res.– 1982.– V. 11, N 1.– P. 42 – 47.
17. Oracle – Hardware and Software docs. – Режим доступу: www.oracle.com/. - Дата доступу: 19.05.2016.
18. Noureddin Sadawi. – Режим доступу: https://www.youtube.com/channel/UCNYv4HA3WjV3gZGLfBehRWQ. - Дата доступу: 27.05.2016.
19. SiteMarker.Ru Интернет технологии. – Режим доступу: http://sitemaker.ru/technologies/database/mysqlvspostgresql/. - Дата доступу: 19.05.2016.
20. Spring Framework Overview. – Режим доступу: http://www.tutorialspoint.com/spring/spring_overview.htm. - Дата доступу: 02.06.2016.
21. Принцип MVC в веб программировании. – Режим доступу:

http://folkprog.net/printsip-mvc-u-web-programmirovanii/. - Дата доступу: 03.06.2016.

22. Spring security. – Режим доступу: http://projects.spring.io/spring-security/. - Дата доступу: 23.04.2016.
23. Hibernate (framework). – Режим доступу: https://en.wikipedia.org/wiki/Hibernate_(framework). - Дата доступу: 13.10.2015.
24. Unit тестирование с JUnit. – Режим доступу: http://devcolibri.com/864. - Дата доступу: 30.05.2016.
25. JUnit. – Режим доступу: https://ru.wikipedia.org/wiki/JUnit. - Дата доступу: 27.04.2016.
26. GoogleApiClient. – Режим доступу: https://developers.google.com/android/reference/com/google/android/gms/common/api/GoogleApiClient. - Дата доступу: 14.04.2016.
27. Google Calendar API. – Режим доступу: https://developers.google.com/google-apps/calendar/. - Дата доступу: 05.06.2016.