Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Факультет \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Комп’ютерних наук\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(повна назва)

Кафедра \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Програмної інженерії\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(повна назва)

**АТЕСТАЦІЙНА РОБОТА**

**Пояснювальна записка**

\_\_\_\_\_\_\_\_\_\_\_\_\_другий (магістерський)\_\_\_\_\_\_\_\_\_\_\_\_\_

(рівень вищої освіти)

Дослідження підходів та можливостей системи формування розкладу занять у \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_навчальному центрі\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(тема)

Виконав: студент 2 курсу, групи ІПЗм-16-1

спеціальності 121 - Інженерія програмного забезпечення\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(код і повна назва спеціальності)

спеціалізації Інженерія програмного \_\_\_\_\_\_\_

забезпечення\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(повна назва спеціалізації)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Пасічник В.Е. \_\_\_\_\_\_\_\_\_\_\_\_

(прізвище, ініціали)

Керівник \_\_\_\_\_\_к.т.н. проф. Білоус Н.В.\_\_\_\_\_\_

(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри \_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_Дудар З.В.\_\_\_\_\_\_\_

(підпис) (прізвище, ініціали)

2018 р.

Харківський національний університет радіоелектроніки

Факультет \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Комп’ютерних наук\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кафедра \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Програмної інженерії \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Рівень вищої освіти\_\_\_\_\_\_\_\_\_\_\_другий (магістерський) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Спеціальність \_\_\_\_\_\_121 – Інженерія програмного забезпечення\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(код і повна назва)

Спеціалізація \_\_\_\_\_\_\_\_\_\_\_\_\_Інженерія програмного забезпечення\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(повна назва)

ЗАТВЕРДЖУЮ:

Зав. кафедри \_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис)

«\_\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 20 \_\_\_ р.

**ЗАВДАННЯ**

НА АТЕСТАЦІЙНУ РОБОТУ

студентові\_\_\_\_\_\_\_\_\_\_\_\_\_*Пасічник Владиславі Едуардівні*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(прізвище, ім’я, по батькові)

1. Тема роботи (проекту*)\_ Дослідження підходів та можливостей системи формування розкладу занять у навчальному центрі*

затверджена наказом по університету від \_16\_ \_квітня\_ 2018 р. № 450 Ст

2. Термін подання студентом роботи до екзаменаційної комісії \_12\_ \_червня\_ 2018 р.

3. Вихідні дані до роботи *Можливості системи формування розкладу занять у навчальному центрі. Підходи формування розкладу. API CIST – сервісу формування розкладу в університеті. Розробити програмну систему для підтримки формування розкладу занять у навчальному центрі. Використовувати технології Node.js, PostgreSQL в якості бази даних, HTML/CSS/Javascript, Angular 2, UML моделювання.*

4. 3міст пояснювальної записки (перелік питань, що потрібно розробити) *мета роботи, аналіз користувальницьких і розробка функціональних вимог до програмного продукту,* *опис прийнятих проектних рішень, методи та алгоритми, що використовувались, опис роботи застосування, тестування ПЗ та аналіз дослідної експлуатації. Додатки: а) наукова публікація, б) довідка про наявність публікації, в) приклади програмних кодів, г) слайди презентації, д) електронні матеріали до проекту на CD.*

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень) *зображення конкуруючих програмних продуктів, схема складників CRM систем, діаграма прецедентів, діаграма розгортання, схема архітектури додатку, приклади програмного коду, інтерфейс веб-системи, слайди презентації*

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1 )

|  |  |  |  |
| --- | --- | --- | --- |
| Найменування  розділу | Консультант  (посада, прізвище, ім’я, по батькові) | Позначка консультанта  про виконання розділу | |
| підпис | дата |
| Снецчастина | проф. Білоус Н.В. |  |  |

**КАЛЕНДАРНИЙ ПЛАН**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Назва eтапів роботи | Терміни  виконання етапів роботи | Примітка |
| 1 | Об'єктний аналіз поставленої задачі | 25-02-2018 | виконано |
| 2 | Аналіз предметної галузі | 12-03-2018 | виконано |
| 3 | Аналіз існуючих рішень | 15-04-2018 | виконано |
| 4 | Дослідження існуючих методів | 28-05-2018 | виконано |
| 5 | Пошук та вирішення недоліків алгоритмів та систем | 15-05-2018 | виконано |
| 6 | Підготовка пояснювальної записки. | 20-06-2018 | виконано |
| 7 | Підготовка презентації та доповіді | 01-06-2018 | виконано |
| 8 | Попередній захист | 04-06-2018 | виконано |
| 9 | Нормоконтроль, рецензування | 07-06-2018 | виконано |
| 10 | Занесення диплома в електронний архів | 07-06-2018 | виконано |
| 11 | Допуск до захисту у зав. кафедри | 08-06-2018 | виконано |

Дата видачі завдання «22» січня 2018 р.

Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис)

Kepiвник роботи \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_проф. Білоус Н.В.\_\_

(підпис) (посада, прізвище, ініціали)

**РЕФЕРАТ / ABSTRACT**

Пояснювальна записка до атестаційної роботи: 84 с., 23 рис., 5 додатки, 25 джерел.

ВЕБ-СИСТЕМА, РОЗКЛАД, НАВЧАННЯ, АВТОМАТИЗАЦІЯ, БАЗИ ДАНИХ, TYPESCRIPT, NODE.JS, ANGULAR, HTML5/CSS3.

Об’єктом дослідження є процес формування розкладу у науковому центр, що працює на базі університету, а саме «Навчально-виробничий центр аутсорсингу», який має навчальні аудиторії на базі університету ХНУРЕ, в яких проводяться заняття як університету, так і власне центру.

Метою роботи є розробка підходу та програмної системи формування розкладу занять у навчальному центрі з підтримкою можливості інтегрування розкладів центру та університету.

Методи розробки базуються на технологіях Node.js, Angular5, HTML / CSS / TypeScript.

У результаті роботи здійснено аналіз підходів формування розкладу та можливостей існуючих CRM систем, порівняні та обрані технології розробки, запропонована модель нового програмного забезпечення для формування розкладу занять у навчальному центрі, що працює на базі університету.

WEB SYSTEM, SCHEDULE, STUDYING, AUTOMATION, DATABASES, TYPESCRIPT, NODE.JS, ANGULAR, HTML5 / CSS3.

The object of research is process of scheduling in a university-based research center, «Outsourcing Training and Production Center», which has educational rooms on the basis of the NURE university, in which classes are held both from the university and from the center itself.

The aim – the development of an approach and program system for classes scheduling in the training center, with the support of the possibility of integrating the schedules of the center and the university.

The development methods are based on Node.js, Angular5, HTML / CSS / TypeScript.

As a result of the work, the approaches to the scheduling and capabilities of existing CRM systems were analyzed, the software development technologies were compared and selected, the model of new software to manage the schedule of classes in the university-based training center was proposed.

**ЗМІСТ**

|  |  |
| --- | --- |
| Перелік скорочень | 6 |
| Вступ | 7 |
| 1 Аналіз предметної області та постановка задачі | 9 |
| 1.1 Аналіз предметної області розробки | 9 |
| 1.2 Аналіз можливостей існуючих систем | 11 |
| 1.3 Постановка задачі | 19 |
| 1.4 Перелік вимог до програмної системи | 21 |
| 2 Дослідження підходів формування розкладу | 23 |
| 2.1 Загальні підходи формування розкладу | 23 |
| 2.2 Алгоритм додавання подій до розкладу навчального центру | 26 |
| 3 UML моделювання та вибір технологій розробки | 30 |
| 3.1 UML – моделювання програмної системи | 30 |
| 3.2 Аналіз та вибір технологій розробки | 34 |
| 4 Опис програмної реалізації | 38 |
| 4.1 Опис функціоналу системи для користувача типу «студент» | 39 |
| 4.2 Опис функціоналу системи для користувача типу «викладач» | 41 |
| 4.3 Опис функціоналу системи для користувача типу «адміністратор» | 49 |
| 5 Тестування та експлуатація системи | 52 |
| 5.1 Тестування розробленої системи | 52 |
| 5.2 Експлуатація системи | 57 |
| Висновки | 59 |
| Перелік джерел посилання | 61 |
| Додаток А Публікація «Деякі проблеми застосування СRM систем для автоматизації діяльності Навчального Центру Аутсорсингу університету» | 63 |
| Додаток Б Лист від оргкомітету XIV Міжнародної конференції «Стратегія якості в промисловості та освіті» | 69 |
| Додаток В Приклади програмного коду | 71 |
| Додаток Г Слайди презентації | 77 |
| Додаток Д Електронні матеріали атестаційної роботи (CD диск) |  |

**ПЕРЕЛІК СКОРОЧЕНЬ**

|  |  |
| --- | --- |
| НВЦА | Навчальний Науково-Виробничий Центр аутсорсингу |
| ХНУРЕ | Харківський національний університет радіоелектроніки |
| CIST | Сервіс розкладу Харківського національного університету радіоелектроніки |
| CRM | Customer Relationship Management |
| NURE | Kharkiv National University pf Radio Electronics |
| OSTPC | Outsourcing Training and Production Center |

**ВСТУП**

Більшість людей все частіше помічають той факт, що завантаженість особистими справами та робочими завданнями настільки велика, що вони просто не знають, як встигнути виконати усі задачі. Від постійного завантаження та нестачі часу, людина стає розсіяною, має поганий настрій та може просто забути про якусь важливу справу. Для запобігання таких проблем існують багато методик тайм-менеджменту, що дозволяють навчитися грамотно планувати свій час.

Одною із простих методик управління часом [1] є складання списку завдань з певними часовими межами. Це дозволяє збільшити ефективність використання як особистого, так і робочого часу. Легко керувати інформацією, коли завжди маєш до неї доступ. У сучасний період стрімкого розвитку інформаційних технологій такі пристрої, як комп’ютер, смартфон чи планшет стали невід’ємною частиною нашого повсякденного життя. Паперові щоденники, що раніше мали більшу популярність, поступово втрачають свою актуальність, бо телефон, що завжди під рукою, може замінити усі їх функції та навіть розширити їх можливості. Стрімко зростає на ринку кількість додатків, що дозволяють керувати своїм щоденним розкладом, синхронізувати його на різних пристроях, групувати справи в залежності від їх типу та ін..

Часто буває так, що для організації якогось заходу треба узгоджувати розклади великої кількості людей. Доволі складно обробити велику кількість інформації самотужки, враховуючи те, що дані динамічні та можуть змінюватися, чи, наприклад, те що треба організовувати декілька подій.

Дослідивши організацію робочого процесу у навчальному центрі та вчителів в університеті, було відзначено, що процес створення робочого розкладу не є повністю автоматизованим. Маючи за основу розклад університетських занять у кабінетах навчального центру, розклад занять самого центру, власний розклад, розклад студентів університету та навчальної лабораторії, викладачу доводиться зіставляти ці розклади, іноді навіть використовуючи їх паперові версії у зв’язку з відсутність електронних Така робота займає багато часу, потребує уваги. Її не зручно виконувати вручну, бо доводиться постійно тримати в голові багато факторів, що впливають на розклади в різних учбових лабораторіях та різних навчальних групах. Отже, ця задача є складною та потребує автоматизації.

Робота має важливе значення для кафедри, так як в університеті діє «Навчально-виробничий центр аутсорсингу» [2], що має навчальні лабораторії, які використовуються в якості аудиторії для студентів університету ХНУРЕ, так і для проведення занять навального центру. Отже, виникає проблема керування розкладами для цих закладів.

Метою атестаційної роботи є розробки підходу та програмної системи для формування розкладу занять у навчальному центрі.

Система є актуальною, бо на даний час відсутня у навчальному центрі. У результаті програмне забезпечення допоможе викладачам центру керувати завантаженістю учбових аудиторій та автоматизувати процес складання та заставлення розкладів класів. Після аналізу можливостей існуючих систем можна з впевненістю сказати, що поставлена задача є новітньою, бо включає в себе інтегрування зі стороннім сервісом та розширює переваги існуючих СRM систем.

У результаті дослідження поточного процесу складання розкладу для начальних лабораторій поставлено за мету створити систему, з використанням новітніх підходів та можливостей формування розкладу занять у навчальному центрі. Робота є актуальною, бо дозоляє напіватоматизувати вищеперераховані процеси, візуалізувати розклади роботи. Раніше подібна система була відсутня.

У результаті проведене дослідження було апробоване в доповіді «Деякі проблеми застосування СRM систем для автоматизації діяльності Навчального Центру Аутсорсингу університету» для XIV Міжнародної конференції «Стратегія якості в промисловості та освіті» (див. додатки А та Б). Приклади коду розробленого додатку наведені у додатку В. Було розроблено презентацію (див. додаток Г). Усі матеріали наведені на диску (див. додаток Д).

**1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ**

1.1 Аналіз предметної області розробки

У сучасному світі, що стрімко розвивається, важливу роль займають процеси автоматизації у всіх галузях життя. Складні виробничі процеси виконуються машинами, що полегшує та зменшує час їх виконання. Технології розповсюджуються і на повсякденне життя: від електронних повідомлень до роботів, що вміють виконувати значну частину домашньої роботи. Такі програмні системи та прилади спрошують процеси керування часом. Складно переоцінити важливість цього питання у наші часи, коли нас оточують великі об’єми інформації, та і ритм життя став швидше. Тому питання тайм менеджменту постає все частіше. Можливість пришвидшити виконання усяких частих процесів, завжди мати доступ до потрібної та актуальної інформації – ось що цікавить сучасну людину.

На жаль, процеси автоматизації розповсюдилися не в усі сфери у повній мірі. Так, наприклад, докладно розглянувши процеси в організації роботи викладачів у навчальному центрі, було виявлено ряд проблем. Основними з них являються:

* відсутність онлайн розкладу занять у лабораторіях начального центру. Викладачу курсу потрібно слідкувати за розкладом лабораторій на сайті університету, координувати графік з іншими викладачами центру. Це займає багато часу, потребую багато зусиль и може призвести до людських помилок в складанні розкладу. Складність цього процесу збільшується з прямою пропорційністю від кількості аудиторій центру, кількості пар у них та кількості студентів;
* можливість змін в розкладі пар університету, що призводить до того, що пари центру потребують повторного переставлення;
* відсутність можливості студентів центру переглядати розклад в онлайн режимі;
* відсутність можливості централізовано керувати розкладом лабораторій з урахуванням кількості студентів на занятті та можливості об’єднувати заняття для одночасно кількох груп з метою оптимізації використання ресурсів;
* відсутність централізованого каналу зв’язку між викладачами центру та студентами для обміну актуальною інформацію, навчальними матеріалами та таке інше.

Враховуючи вищезазначені проблеми, постає питання про те, як можна полегшити ці процеси. Основним питанням є можливість інтеграції з СIST (сервісом розкладу ХНУРЕ). Отже, виникають складності у використанні існуючих сервісів. Тому виникає наочна необхідність у створенні власної програмної системи, що буде враховувати усі вищеперераховані завдання.

Час – це дуже цінний, непоновлюваний ресурс. Те як ми розпоряджаємося цим ресурсом визначає долі людей і організацій. Тайм-менеджмент – це технологія організації часу і підвищення ефективності його використання. Тайм-менеджемент це не стільки набір технік, скільки стиль життя і філософія цінності часу в швидкому потоці інформації і постійно мінливому світі.

Базові ідеї управління часом полягають в тому, що основну увага потрібно приділяти тому, куди має інвестуватися час, а не тому, як швидше робити справи. По суті, це дуже важливе і складне завдання постановки цілей, визначення пріоритетів, яка зачіпає всі галузі управлінської діяльності, пов'язані з прийняттям рішень.

Для вирішення задачі зіставлення учбових пар та завантаження аудиторій слід звернути увагу на теорію розкладів та алгоритми керування часом. Теорія розкладів є однією з наук, що займається завданнями дискретної оптимізації. Для неї характерно велике розмаїття теоретичних моделей, що випливає з різноманітності реальних модельованих процесів. (Легко зрозуміти, що проблема оптимальної організації протікають у часі процесів має глобальний характер і виникає практично у всіх сферах людської діяльності.) Другою особливістю є комбінаторна природа досліджуваних моделей і розв'язуваних оптимізаційних задач, яка обумовлює високу комбінаторних складність їх вирішення. І, по-третє, для теорії розкладів характерна висока актуальність вирішуваних завдань, з огляду на їх яскраво виражену прикладну спрямованість.

Отже, у результаті проведеного аналізу предметної області було вирішено дослідити підходи та можливості системи формування розкладу занять у навчальному центрі та на основі проведених досліджень розробити програмну систему, для керування учбовим процесом у центрі.

1.2 Аналіз можливостей існуючих систем

У даний час на ринку існує велика кількість СRM систем, які можна використовувати для керування процесами у навчальному центрі. Але провівши аналіз ряду систем, було зроблено висновок, що у зв’язку з усіма потребами та особливостями навчального центру OSTPC, існуючи програмні засоби не можуть бути використані. Далі детально розглянемо та проаналізуємо CRM системи, обрані для дослідження.

CRM система (Customer Ralationship Management) [3] – це прикладне програмне забезпечення для організацій, призначене для автоматизації стратегій взаємодії з замовниками (клієнтами), зокрема, для підвищення рівня продажів, оптимізації маркетингу і поліпшення обслуговування клієнтів шляхом збереження інформації про клієнтів та історії взаємин з ними, встановлення і поліпшення бізнес-процесів, подальшого аналізу результатів.

CRM - модель взаємодії, що визначає: центром всієї філософії бізнесу є клієнт, а основними напрямками діяльності є заходи з підтримки ефективного маркетингу, продажу та обслуговування клієнтів. Підтримка цих бізнес-цілей включає збір, збереження та аналіз інформації про споживачів, постачальників, партнерів, а також про внутрішні процеси компанії. Функції для підтримки цих бізнес-цілей включають продажі, маркетинг, підтримку споживачів.

CRM системи класифікуються за функціональними можливостями та за рівнями обробки інформації. За функціональними можливостями вони поділяються на системи управляння продажами, управляння маркетингом, управління клієнтським обслуговуванням. За рівнем обробки інформації йде наступний поділ:

* операційний CRM - реєстрація та оперативний доступ до первинної інформації щодо подій, компаній, проектів, контактів;
* аналітичний CRM - звітність і аналіз інформації в різних розрізах (воронки продажів, аналіз результатів маркетингових заходів, аналіз ефективності продажів в розрізі продуктів, сегментів клієнтів, регіонів і інші можливі варіанти);
* коллаборативний CRM - рівень організації тісної взаємодії з кінцевими споживачами, клієнтами, аж до впливу клієнта на внутрішні процеси компанії (опитування, для зміни якостей продукту або порядку обслуговування, веб-сторінки для відстеження клієнтами стану замовлення, повідомлення по SMS про події, пов'язані із замовленням або особовим рахунком, можливість для клієнта самостійно конфігурувати і замовляти в режимі реального часу продукти та послуги).

Вибір CRM-систему є важкою задачею, зважаючи на їхню різноманітність на ринку. Під час вибору CRM системи треба враховувати усі поточні потреби бізнесу. Детально розглянувши ринку, було виявлено, що найбільш популярними на даний час CRM системами є нижче проаналізовані додатки. Слід зазначити, що особлива увага буде приділятися можливості інтегрування модулю розкладу в CRM системі з розкладами з сторонніх сервісів, так як основною задачею для навчального центру OSTPC на базі університету є можливість впровадити розклад ВУЗу до свого власного.

Prosperworks [4] – CRM система, що була створена для Google, використовується та рекомендована ним. Основними її можливостями є:

* легке інтегрування з Google таблицями, документами, презентаціями, календарем, поштою, тому ії використання просте, бо з Google інструментами знайомий майже кожен і всі дані зберігаються в одному місці;
* легкий доступ до списку контактів та інтеграція з різноманітними популярними сервісами для імпортування контактних даних;
* отримання інформативних звітів, виявлення критичних проблем бізнесу, які підвищать ефективність команди та оптимізуватимуть потік маркетингу;
* висок надійність та збереження даних.

На рисунку 1.1 приведено інтерфейс модулю для керування інформацією про компанію, що надається CRM системою Prosperworks.

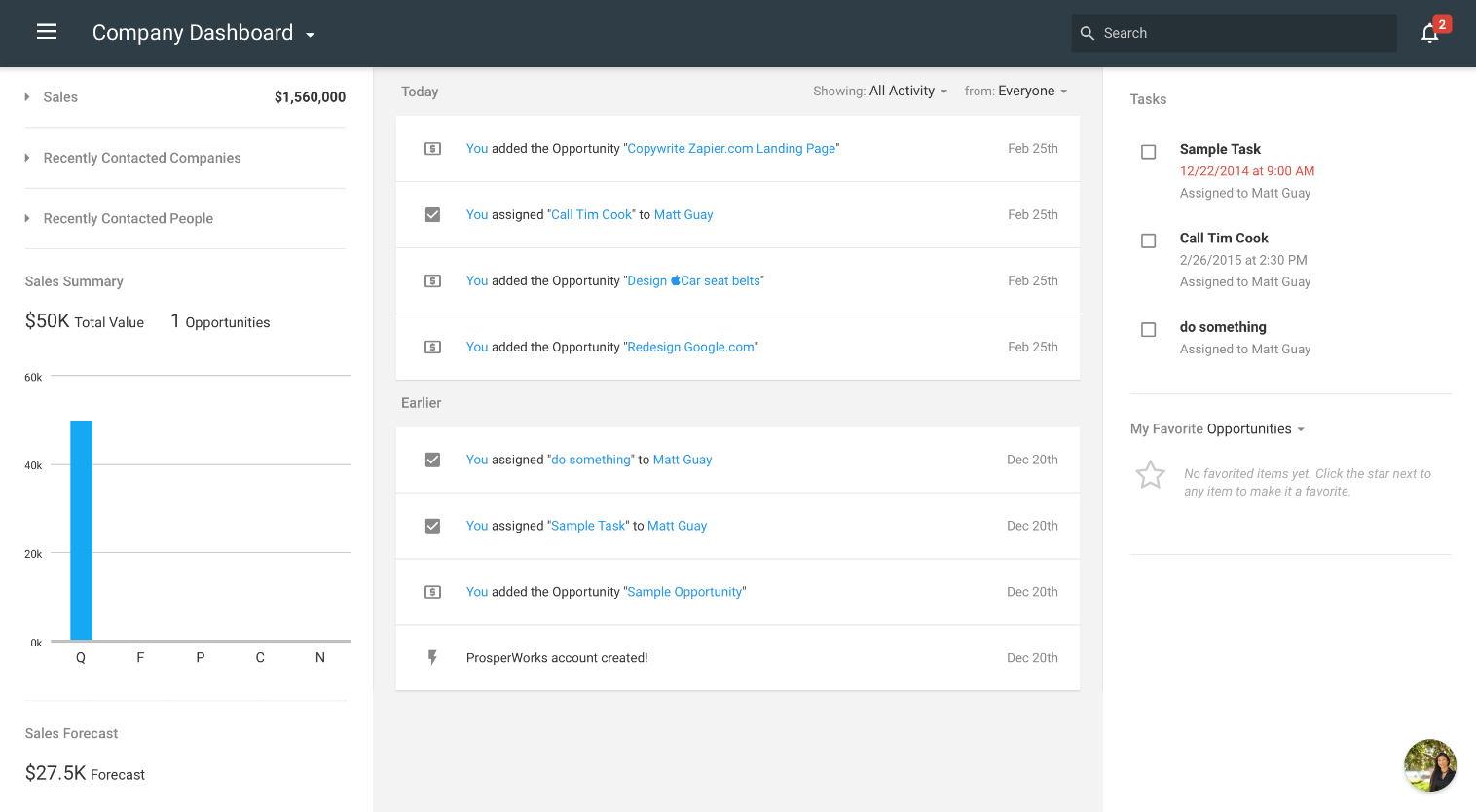


Рисунок 1.1 – Інтерфейс модулю компанії від CRM Prosperworks

Вище перераховані переваги Prosperworks роблять дану CRM систему досить привабливою для впровадження в роботу навчального центру «Навчально-виробничого центру аутсорсингу». Але як ми бачимо, у системі має можливостей для розширення та інтегрування з розкладами зі сторонніх сервісів. Також слід звернути увагу на те, що сервіс є платним та розрахований на більш великі команди.

ZohoCRM [5] - супермен в світі управління взаємовідносинами з клієнтами на 3-х ключових рівнях: маркетинг, продажі, support. Сервіс дозволяє контролювати і автоматизувати основні бізнес-процеси. Функціонал: кроссплатформеність; Інтеграція з Facebook, Twitter, Gmail, WordPress; перевірка відвідувачів сайту і сегментація; відстеження джерел трафіку; історія взаємодій, звіти, повідомлення; управління клієнтською базою; клієнтський сервіс; зручний імпорт / експорт даних; зручний розклад завдань, календар; планування, прогноз, аналіз воронки. ZohoCRM (див. рис. 1.2) дозволяє зібрати всі важливі дані в одному місці: контакти покупців, їх історію, інформацію про джерела трафіку, кількості продажів і т.д.

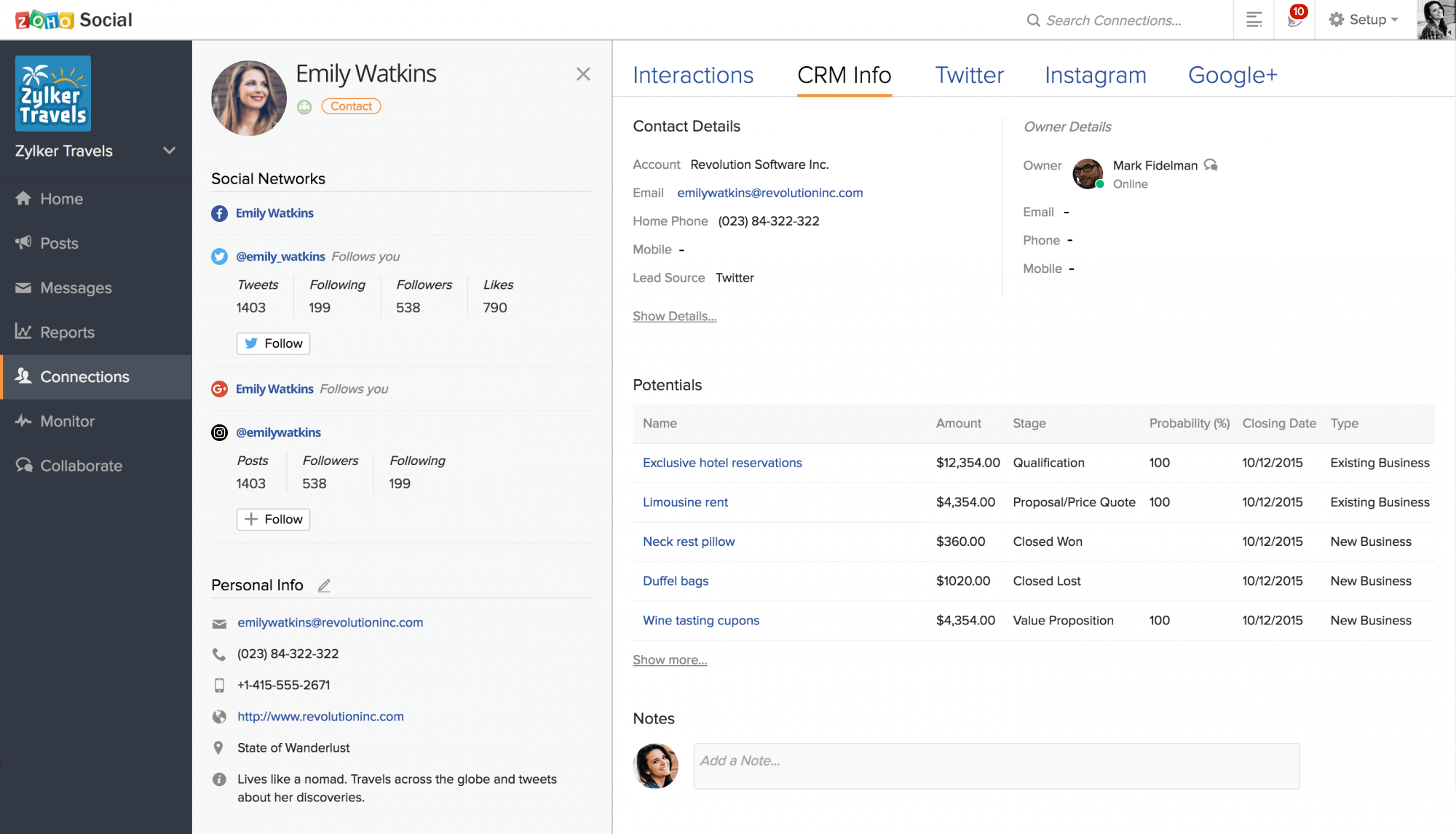


Рисунок 1.2 – Екранна форма CRM системи ZohoCRM

AmoCRM [6] - це хмарний сервіс, формату SFA (Sales Force Automation - система автоматизації продажів). Простий і зрозумілий інтерфейс дозволяє без особливих зусиль налаштовувати і користуватися нею через веб-додаток і з мобільних пристроїв. Сервіс пропонує: створення клієнтської бази, збереження історії взаємодії з кожним клієнтом; контроль потоку заявок, інформація по кожній угоді; аналітика: аналіз продажів, зведений звіт, звіт по співробітниках; API - готові рішення (веб-форми, віджети для сайту і т.д; тайм-менеджмент, планувальник завдань, нагадування, звіти; зручний імпорт і експорт важливих даних.

Бітрікс24 [7] - зручна, багатофункціональна, а також безкоштовна система, яка вже кілька років не здає позиції в рейтингу CRM-систем. Основний упор у сервісі зроблений саме на управління проектами, завданнями, документацією, звітами, планами. Разом з цим Бітрікс24 (див. рис. 1.3) непогано справляється з оптимізацією відносин з клієнтами.

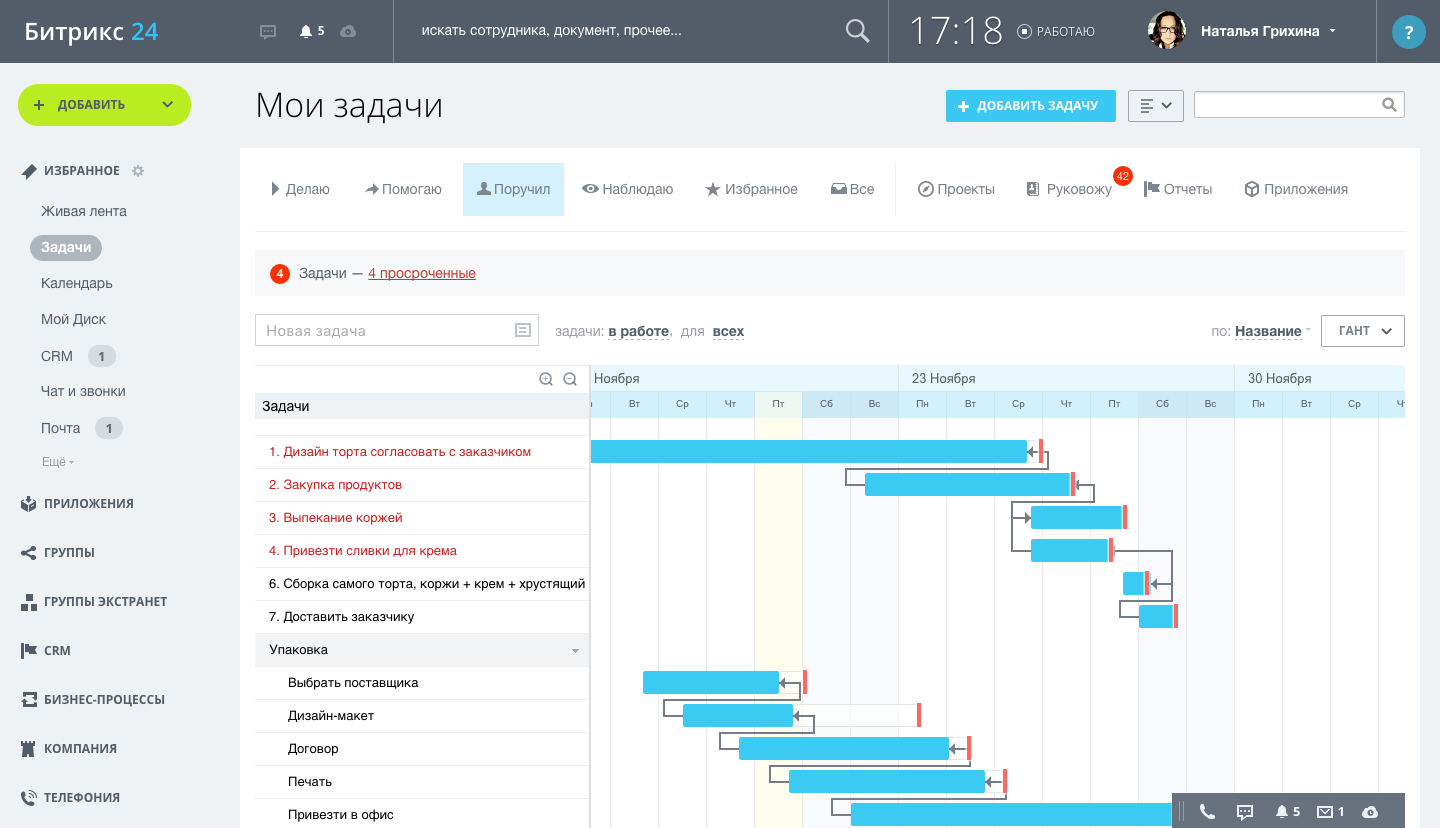


Рисунок 1.3 – Інтерфейс Бітрікс24

На відміну від попередніх 2-х систем, Бітрікс розрахований на впровадження у всю компанію, а не тільки в відділ продажів. Основні функції: ведення та управління базою даних клієнтів; фіксація різних взаємодій (дзвінки, листи); робота з документами та фінансами; відстеження замовлень, оплат; кроссплатформеність; внутрішній чат, розклад, календар, постановник завдань, стрічка змін по проектам, база знань, бейджи; управління проектами; email-трекер - збереження історії листування всередині системи; інтеграція з соцмережами; технічна підтримка. Цей сервіс оптимізує взаємодію між різними відділами, автоматизує робочі.

В огляді CRM-систем для малого бізнесу, а також для малобюджетних стартапів, Бітрікс24 - безперечний лідер як мінімум по одній важливій причині - сервісом можна користуватися безкоштовно, якщо в ньому задіяно не більше 12 осіб.

BaseCRM [8] - система, за допомогою якої можна відслідковувати продажу, угоди, прогнозувати майбутні доходи, складати вичерпні звіти. З іншого боку - це потужна машина для підвищення продуктивності працівників компанії (планинг, KPI, звіти по ефективності, постановка цілей / завдань). А також багаторівневий контроль даних по клієнтській базі (контакти, угоди, листи, документи, історія торкань). На рисунку 1.4 представлено модулю розкладу у BaseCRM.



Рисунок 1.4 - Модулю розкладу у BaseCRM

Функціонал сервісу: наскрізні звіти про розвиток конкретної угоди; аналіз ефективності співробітників; аналітика воронки продажів; прогнозування динаміки продажів; інформаційні картки клієнтів; вбудована телефонія; робота з документами, фінансами; постановка завдань, календар.

Salesforce Sales Cloud [9] - хмарна система планування, управління і контролю бізнес-процесів (відносини з клієнтами, маркетинг, бюджет, продажу, аналітика). Функції: таргетинг відвідувачів; внутрішній онлайн-чат для менеджерів; облік / робота з угодами, оплатами; хмарне сховище даних; управління клієнтською базою; звіти про взаємодію з покупцями; автоматичний розподіл заявок між менеджерами; інтеграція з соцмережами, поштовими сервісами; доступ до сервісу з мобільних пристроїв; призначення пріоритетності заявками; формування робочих завдань, планування.

Отже, основні модулі СRM систем (див. рис. 1.5) можна об’єднати у наступні категорії:

* Customer Service & Support (CSS), або автоматизація служби підтримки та обслуговування клієнтів
* Sales Force Automation (SFA), або автоматизація діяльності продавців;
* Marketing Automation (MA), або автоматизація маркетингу.



Рисунок 1.5 – Модулі СRM систем

Згідно специфіки роботи навчального центру, що з одного боку займається навчальною діяльністю на базі університету, а з іншого діяльність за межами ВУЗу (проведення курсів, трененгів, семінарів, пошук студентів, рекламна діяльність і т.і.), особливу уваг при виборі CRM системи слід звернути на CSS модуль. При аналізі даного модулю у різних вищеперерахованих та інших присутніх на ринку CRM системах, особливу вагу було приділено задачі складання розкладу. Майже усі системи дають таку функцію. Але не було знайдено можливості інтеграції за сторонніми сервісами для розширення цієї саме задачі.

Навчальний центр OSTPC знаходиться на базі ХНУРЕ. Тому його учбові аудиторій частково задіяні для учбового процесу та поза навчальних заходів університету. Отже постає питання про додання пар центру до розклад. Розклад ХНУРЕ доступний до перегляду у системі CIST [10]. На рисунку нижче (див. рис. 1.6) представлений скріншот сторінки CIST для отримання розкладу навального процесу в ХНУРЕ.

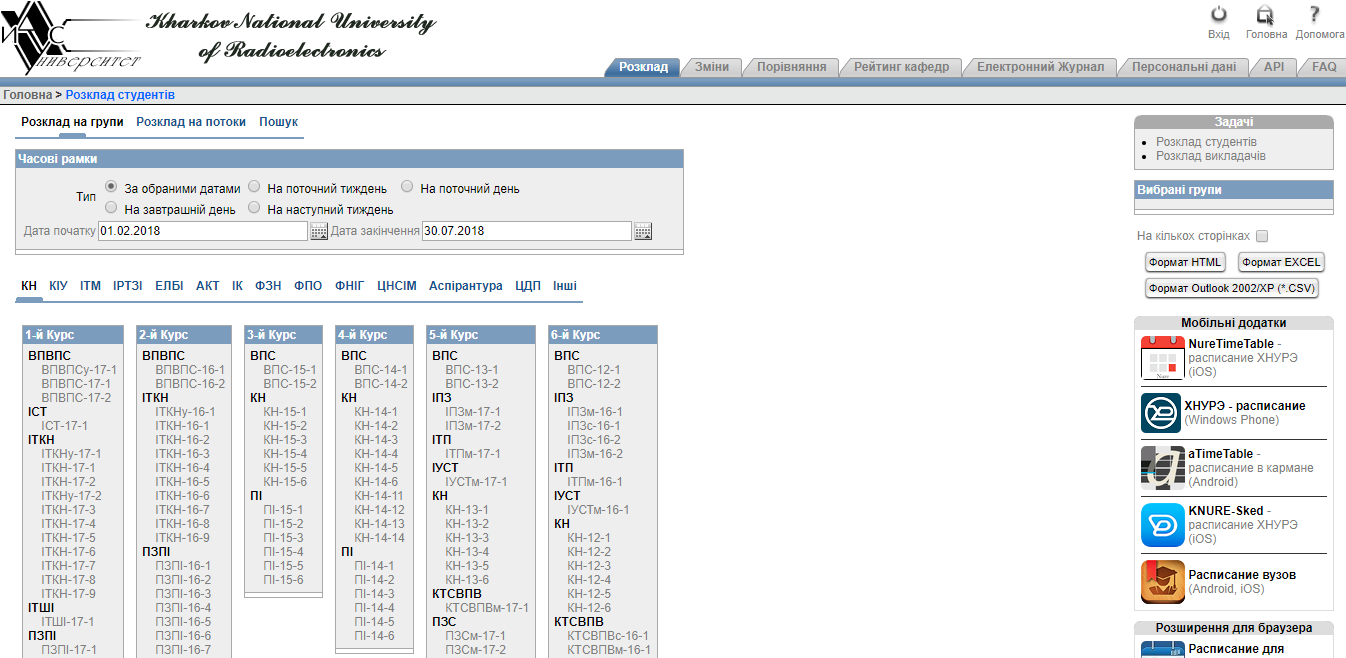


Рисунок 1.6 – Сторінка розкладу сервісу CIST

Даний портал надає доступ до розкладу університету, електронного журналу, а також має відкрите API для можливості інтегрування з сервісом. Простим рішенням було б додавати пари центру до сітки розкладу CIST. Але це рішення не є прийнятним у зв’язку зі специфіко роботи центру. По-перше, центр має гнучкий графік пар, а миттєве додання через навчальний центр не можливе. По-друге, сайт розкладу університету не надає можливості вибрати англійську мову, що робить пошук пар для неукраїномовних студентів центру – важким. Інтерфейс СIST також не відповідає сучасним вимогам та не заохочує до використання. По-третє, на сайті СIST на додаються розклади не учбових заходів, які також необхідно враховувати при плануванні розкладу OSTPC.

Отже, враховуючи специфіку потреб навчального центру OSTPC інтегруватися та розширювати систему розкладу ХРУРЕ CIST та відсутність таких можливостей у готових рішеннях, постає питання про реалізацію власного програмного продукту.

1.3 Постановка задачі

Метою атестаційної роботи є розробка підходу та програмного забезпечення для формування розкладу у навчальному центрі, призначеної для додавання пар у розклад, управління завантаженістю лабораторії, спілкуванню зі студентами. Для досягнення мети було проведено аналіз предметної області, підходів та можливостей сучасних CRM систем, також попередньо потрібно провести UML-моделювання.

Після проведення відповідних досліджень, створення програмної системи складається з реалізації наступних задач:

– реєстрація користувачів у системі (враховуючи різні ролі);

– перегляд розкладу занять;

– інтеграція системи з розкладом CIST для відповідних аудиторії, груп та викладачів;

– надання можливостей студенту: переглядати розклад, надсилати повідомлення викладачу ;

– надання можливостей викладачу: переглядати розклад занять аудиторії, надсилати запити на додання пар до розкладу аудиторії з перевіркою на перетин з розкладом студента (з рядом критеріїв, таких як час проведення, пари, кількість студентів, зайнятість комп’ютерів та ін.), спілкування зі студентами та адміністратором, отримувати сповіщення про накладання пар;

– надання можливості адміністратору: приймати/відхиляти запити на додання пар до розкладу, додавати викладачів до системи, отримувати сповіщення про накладання розкладу, бачити різні вибірки і звіти по завантаженості аудиторій, а також зведений розклад кількох академічних груп, наприклад, за фахом, роком навчання, факультетом; вказувати наявності в їх аудиторії унікальних навчальних ресурсів (наприклад, встановленого програмного забезпечення, проектора, кількості посадочних місць), які розробники курсів можуть вказувати як необхідну для проведення занять; внесення до розкладу ненавчальних заходів з перевіркою їх перетинів з розкладом;

– можливість отримувати інформацію про стан завантаженості аудиторії для оптимального використання ресурсів.

Для реалізації системи було обрано середовище розробки WebStrom [11]. Так як засіб буде розроблено у вигляді веб-системи це забезпечить кросплатформеність та можливість роботи у будь якій системі – з сімейства Windows або Unix-подібних систем. Мовою написання серверної частини був обраний Node.js з використанням веб-фреймворку express. В якості сховища даних була обрана СУБД PostgreSQL. Для реалізації клієнтської частини обрані технології HTML5/CCS3, TypeScript, з використанням фреймворку Angular5 та бібліотеки AngularMaterial2.

У ході виконання роботи необхідно зробити такі завдання:

– провести аналіз, концептуальне та UML-моделювання предметної галузі;

– провести дослідження існуючих підходів та можливостей систем формування розкладу;

– розробити схему бази даних для зберігання усіх даних користувача.

1.4 Перелік вимог до програмної системи

Перед початком роботи над програмною системою необхідно визначитися з вимогами до неї. У загальному випадку під вимогами розуміють сукупність властивостей, які повинна мати система, що реалізується. У ході роботи треба реалізувати програмну систему для полегшення формування розкладу занять у навчальному центрі.

Даний додаток призначений для викладачів навчального центру (на даний час саме для Учбового науково-виробничого центру аутсорсингу). Дана система має сучасний інтерфейс, тому не потребує багато часу для навчання користування, і , як наслідок, навпаки зменшує час, що потрібен користувачу (викладачу) для внесення пар до розкладу навчального центру. Інтегрування розкладу CIST у додаток, слідкування за завантаженістю аудиторії, да можливість максимально оптимізувати роботу у навчальному залі. Сповіщення про накладання пар – допомагає запобігти конфліктам в розкладах. Таким чином зникає необхідність постійного перегляду розкладів аудиторії занять у вузі та ручного порівняння з заняттями у навчальному центрі, що значно полегшує життя викладачеві, зменшуючи кількість зайвої роботи.

Загальні відомості про розробку системи полягають у наступному: система формування розкладу занять у навчальному центрі має бути розроблено за допомогою мови програмування Node.js, фреймворку express. та технологій HTML5/CCS3, ТуреScript, з використанням фреймворку AngularJS та бібліотеки Bootstrap.

В якості сховища даних була обрана СУБД PostgreSQL [12]. Засіб було вирішено створити у вигляді веб-додатку (сайту), що підтримує роботу з чотирма типами користувачів: викладачем ХНУРЕ, зовнішнім викладачем, студентом, адміністратором. При перегляді сайту, користувач має змогу переглянути розклади аудиторії. Після вводу свого логіну і паролю вчителям та адміністратору надається доступ до повного функціоналу системи.

Основна функціональність даної системи полягає у наступному:

– перегляд розкладу занять в аудиторіях;

– зіставлення розкладу занять університету та навчального центру;

– комунікація зі студентами.

При створенні підсистеми повинно бути взято до уваги:

– зручність використання сайту – інтерфейс має бути зручним та інтуїтивно зрозумілим, не вимагати додаткової підготовки користувачів;

– довідкова інформації щодо використання системи повинна бути розташована у відповідному розділі на сайті.

Система розробляється у вигляді веб-додатку, тож вона не потребує специфічних умов для того, щоб користувач зміг працювати з нею. Серед необхідних вимог для початку роботи можна назвати лише:

– комп’ютер з підключенням до мережі Інтернет;

– наявність програми для перегляду веб-сторінок – браузеру.

**2 ДОСЛІДЖЕННЯ ПІДХОДІВ ФОРМУВАННЯ РОЗКЛАДУ**

2.1 Загальні підходи формування розкладу

Складання розкладу - одна з найбільш поширених завдань в плануванні і оптимізації навчального процесу в начальних закладах. Від того, наскільки добре складений розклад, залежить ефективність роботи викладача, засвоєння навчального матеріалу студентами, раціональне використання ресурсів навчального закладу. Автоматизація складання розкладу - досить класична задача в системах управління навчальним процесом, але на даний момент немає єдиного, загальноприйнятого способу її вирішення.

Очевидний підхід скласти розклад, розглядаючи всі можливі варіанти (метод повного перебору), можливий, коли математична задача, отримана після формалізації поставленого завдання, має порівняно невелику розмірність. Завдання знаходження оптимального розкладу занять [13] в більшості випадків відноситься до класу складних завдань. Якщо враховувати реальні умови, то завдання ще більш ускладняється, оскільки шукані рішення повинні задовольняти великій кількості обмежень виробничого, організаційного та психофізіологічного характеру, які суперечать один одному. При цьому оптимальне рішення може не існувати. Для подолання цієї ситуації потрібно знайти компроміс між критеріями, тобто скласти згортку критеріїв в один узагальнений критерій. Якщо такий підхід не дає оптимального рішення, слід відмовитися від пошуку оптимального рішення. В якості вирішення задачі в цьому випадку слід виділити не єдине рішення, а деяку множину прийнятних варіантів, з якої вибрати остаточний варіант рішення.

У загальній постановці завдання складання розкладу є процес розподілу деякого кінцевого набору подій у часі в умовах ресурсних та інших обмежень. В нашій задачі в якості вихідних даних для складання розкладу виступають зайнятість викладачів у ВУЗі, у навчальному центрі, зайнятість аудиторій як університетом так і навчальним центром, завантаженість аудиторій, кількості навчальних пар.

Складання розкладу відноситься до завдань цілочисельного програмування, складність рішення яких зростає експоненціально з ростом числа і можливих значень варійованих змінних (такі завдання входять до класу NP-складних завдань). Існують підходи складання розкладу, засновані на так званих точних (класичних) методах і алгоритмах цілочисельного програмування, на методах розмальовування графів. Крім того, застосовуються методи повного перебору, гілок і меж, а також евристичні методи, в тому числі засновані на генетичних алгоритмах.

Завдання складання розкладу можна розглядати як задачу розмальовки графа [14]. Завданням розмальовки графа називають пошук мінімального числа кольорів (хроматичного числа графа), необхідних для розфарбування вершин деякого графа так, щоб кожна пара сусідніх вершин була пофарбована в різні кольори. Сама задача пошуку хроматичного числа є NP-повною задачею, для вирішення якої в більшості випадків використовуються різні жадібні алгоритми.

Для постановки задачі складання розкладу як завдання розмальовки графа будується граф. Кожна вершина графа - це заплановане навчальним планом заняття. Якщо між якимись двома вершинами можливі конфлікти, наприклад, обидва заняття проводяться в одній аудиторії або з одним викладачем, то вони з'єднуються ребром, що еквівалентно забороні одночасного проведення цих занять. Завдання складання розкладу може бути сформульована як задача мінімізації числа кольорів, необхідних для розмальовки графа. Кожен колір відповідає одному періоду розкладу. Завдання розмальовки графа як самостійна є мало ефективною при складанні розкладів, але може виявитися корисною в разі її комбінації з іншими алгоритмами.

Складання розкладу можна розглядати як задачу задоволення обмежень, для вирішення яких розроблено безліч алгоритмів. Виник цілий напрямок в програмуванні - програмування в обмеженнях (constraint programming). Програмування в обмеженнях тісно пов'язане з традиційним логічним програмуванням, в рамках якого воно і сформувалося. Більшість систем програмування в обмеженнях є звичайним інтерпретатор Прологу з вбудованим механізмом для вирішення певного класу задач задоволення обмеженням. Програмування в таких системах називають логічним програмуванням в обмеженнях (Constraint Logic Programming або CLP). Основна ідея вирішення завдань така: програміст визначає деяку множину змінних x1,. . . , xn і області їх значень X1,. . . , Xn, описує додаткові обмеження, яким повинні задовольняти змінні, а система знаходить відповідні значення змінних, що задовольняють одночасно всім заданим обмеженням.

Наведені вище методи в основному використовують ітераційну техніку поліпшення результатів. При виконанні однієї ітерації вони шукають рішення, що є кращим в межах поточного рішення, отриманого на попередній ітерації. Якщо таке рішення знайдено, воно стає поточним і починається нова ітерація. Це продовжується до тих пір, поки не виконається правило зупинки: приріст цільової функції не зменшиться практично до нуля або не виконається задана кількість ітерацій. Очевидно, такі методи здійснюють пошук тільки локальних оптимумів, причому положення знайденого оптимуму залежить від стартової точки, а глобальний оптимум може бути знайдений тільки випадково. Час пошуку при цьому істотній. Тому становить інтерес розробка алгоритмів, які зберігають переваги описаних методів і вільні від зазначеного недоліку. До таких алгоритмам відносяться генетичні алгоритми.

Генетичні алгоритми [15] - це стохастичні евристичні оптимізаційні методи, основна ідея яких була взята з теорії еволюційного розвитку видів. Першим кроком при розробці математичної моделі, заснованої на генетичному алгоритмі, є розробка структури хромосоми, в якій буде зберігатися рішення. У нашому випадку такою «хромосомою» є розклад. Обрана структура повинна враховувати всі особливості і обмеження, що пред'являються до шуканого рішення, а також те, що від її вибору напряму залежать реалізації алгоритмів кросинговеру і мутації. В остаточному підсумку, вибір «хромосоми» впливає не тільки на швидкість, а й на збіжність алгоритму взагалі. Одним з найбільш зручних рішень даної задачі є тривимірна матриця, по осях i, j, k якій відкладаються відповідно групи, навчальні години і аудиторії. Елементом матриці є запит на проведення заняття з i - тою групою даними викладачем з даної дисципліни в j - ту годину в k - тії аудиторії. Кожна зв'язка дисципліна-група-викладач розглядається як єдине ціле за умови, що в алгоритм введена інформація: які дисципліни, в якій групі, який викладач і який тип аудиторій для цього підходить. Така структура хромосоми зручна тим, що вже на етапі завдання початкових даних можна виключити свідомо невдалі рішення, заблокувавши відповідні комірки.

На наступному кроці алгоритму створюється початкова популяція, розмір якої залежить від розмірності задачі і становить зазвичай кілька сотень рішень. Для організації оптимізуючого процесу необхідно створити спрямовуючу силу розвитку популяції. В якості такої сили виступає вимога мінімізації цільової функції (в термінах генетичних алгоритмів, - фітнес-функції). Як фітнес-функцію можна використовувати адитивний показник оптимальності, заснований на штрафах, встановлюваних кожному рішенню за будь-який незручний момент в розкладі. Важлива особливість такого вибору - можливість налаштувати алгоритм на вирішення конкретного завдання. Це досягається варіюванням коефіцієнтів, що призводить до зміни пріоритетів при пошуку оптимального розкладу.

За допомогою даних методів можливе отримання точної математичної моделі, що відповідає всім обмеженням, але в силу NP-складного характеру завдання складання розкладу ця модель буде громіздкою і складною. Щоб уникнути цього, застосовується імітаційне моделювання. У цьому випадку алгоритм оперує безпосередньо розкладом та списком занять, які необхідно включити в розклад.

2.2 Алгоритм додавання подій до розкладу навчального центру

Навчальний центр OSTPC знаходиться на базі ХНУРЕ. Тому його учбові аудиторій частково задіяні для учбового процесу та поза навчальних заходів університету. Отже постає питання про додання пар центру до розклад. Розклад ХНУРЕ доступний до перегляду у системі CIST. Даний портал надає доступ до розкладу університету, електронного журналу, а також має відкрите API для можливості інтегрування з сервісом. Простим рішенням було б додавати пари центру до сітки розкладу CIST. Але це рішення не є прийнятним у зв’язку зі специфіко роботи центру. По-перше, центр має гнучкий графік пар, а миттєве додання через навчальний центр не можливе. По-друге, сайт розкладу університету не надає можливості вибрати англійську мову, що робить пошук пар для неукраїномовних студентів центру – важким. Інтерфейс СIST також не відповідає сучасним вимогам та не заохочує до використання. По-третє, на сайті СIST на додаються розклади не учбових заходів, які також необхідно враховувати при плануванні розкладу OSTPC.

При додаванні заходу до розкладу навчального центру викладачу потрібно враховувати ряд обмежень, а саме:

– захід, що додається, не повинен перетинатися з іншою парою, що проводиться у даній аудиторій;

– захід, що додається, не повинен перетинатися з іншою університетською подією, що відсутня у розкладі центру та що проводиться у даній аудиторій;

– захід, що додається, не повинен перетинатися з парою, що проводить викладач (у разі якщо викладач працює в університеті ХНУРЕ) на базі університету;

– захід, що додається, не повинен перетинатися з парою, що проводить викладач (у разі якщо викладач працює в університеті ХНУРЕ) власне у навчальному центрі;

– у разі якщо захід додається для студентів університету ХНУРЕ, то захід не повинен перетинатися у даний час з учбовими парами для студентських груп.

Враховуючи усі вищеперераховані обмеження було сформовано алгоритм перевірки розкладів на накладання. Обмеження були пріоритезовані (як зазначено у спису обмежень вище). Кожне обмеження послідовно перевіряється, алгоритм припиняється, якщо одне з обмежень не виконується. Викладач отримує сповіщення про те, яке саме обмеження не було виконане у вигляді інформаційного повідомлення. Усі вищеперераховані обмеження відносяться до жорстких обмежень і не можуть бути опущені, при перевірки можливості додати захід до розкладу навчального центру. Нижче приведено код перевірки можливості відправити запит на додання заходу, використовуючи усі вищеперераховані обмеження:

**sendAddRequest() {**

**const scheduleRequestFormValue = this.scheduleRequestForm.value;**

**const startTime = this.formUnixTimestamp(scheduleRequestFormValue.startDate, scheduleRequestFormValue.startTime);**

**const endTime = this.formUnixTimestamp(scheduleRequestFormValue.endDate, scheduleRequestFormValue.endTime);**

**this.areStartEndTimeInvalid = !this.checkStartEndTimestampsValidity(startTime, endTime);**

**if (this.areStartEndTimeInvalid) return;**

**this.cistService.getTeacherSchedule(scheduleRequestFormValue.teacherId)**

**.subscribe((teacherSchedule) => {**

**this.collisionsWithTeacher = this.checkForCollisions(teacherSchedule.events, startTime, endTime);**

**this.collisionsWithTeacherCenter = this.checkForCollisions(this.teacherCenterEvents, startTime, endTime);**

**});**

**this.cistService.getAuditorySchedule(scheduleRequestFormValue.auditoryId)**

**.subscribe((auditorySchedule) => {**

**this.collisionsWithAuditory = this.checkForCollisions(auditorySchedule.events, startTime, endTime);**

**if (scheduleRequestFormValue.auditoryId === ROOM455\_ID) {**

**this.collisionsWithAuditoryCenter = this.checkForCollisions(this.auditory455aCenterEvents, startTime, endTime);**

**}**

**if (scheduleRequestFormValue.auditoryId === ROOM144\_ID) {**

**this.collisionsWithAuditoryCenter = this.checkForCollisions(this.auditory114iCenterEvents, startTime, endTime);**

**}**

**});**

**if (scheduleRequestFormValue.checkUniversityGroupSchedule && scheduleRequestFormValue.universityGroupId) {**

**if (!parseInt(scheduleRequestFormValue.universityGroupId, 10)) {**

**const checkIfGroupIsFromList = this.selectedGroupsList.find((group) =>**

**group.name.toLowerCase() === scheduleRequestFormValue.universityGroupId.toLowerCase()**

**);**

**if (checkIfGroupIsFromList) {**

**scheduleRequestFormValue.universityGroupId = checkIfGroupIsFromList.id;**

**this.scheduleRequestForm.get('universityGroupId').setValue(checkIfGroupIsFromList.id, {emitEvent: false});**

**this.groupIsNotFromTheList = false;**

**} else {**

**this.groupIsNotFromTheList = true;**

**}**

**}**

**if (!this.groupIsNotFromTheList) {**

**this.cistService.getGroupSchedule(scheduleRequestFormValue.universityGroupId)**

**.subscribe((groupSchedule) => {**

**this.collisionsWithGroup = this.checkForCollisions(groupSchedule.events, startTime, endTime);**

**});**

**}**

**}**

**if (this.isFormValid()) {**

**const request = this.formNewEventObject();**

**localStorage.setItem(**

**`request-${scheduleRequestFormValue.teacherId}-${startTime}-${endTime}`,**

**JSON.stringify(request));**

**this.snackBar.open('Your request was successfully send', 'Ok');**

**this.dialogRef.close(request.events);**

**}**

**}**

Після успішної перевірки усіх обмежень, запит викладача на додання заходу відправляється на розгляд адміністратором сайту.

**3 UML МОЕЛЮВАННЯ ТА ВІБІР ТЕХНОЛОГІЙ РОЗРОБКИ**

3.1 UML – моделювання програмної системи

Моделювання розроблюваної системи проводиться з використанням мови UML для побудови діаграм, що допоможуть відобразити функціональність та внутрішню структуру системи. UML - мова графічного опису для об'єктного моделювання в області розробки програмного забезпечення. UML [16] є мовою широкого профілю, це відкритий стандарт, який використовує графічні позначення для створення абстрактної моделі системи, що називається UML-моделлю.

Можна виділити наступні переваги UML:

– UML об'єктно-орієнтований, в результаті чого методи опису результатів аналізу і проектування семантично близькі до методів програмування на сучасних об'єктно-орієнтованих мовах;

– діаграми UML порівняно прості для читання після досить швидкого ознайомлення з його синтаксисом;

– UML розширює і дозволяє вводити власні текстові та графічні стереотипи.

При проектуванні системи були розроблені діаграми наступних типів: діаграма прецедентів та діаграма розгортання.

Діаграма прецедентів описує функціональне призначення системи або те, що система повинна робити. Розробка діаграми має такі цілі:

– визначити контекст предметної області, яка моделюється;

– сформулювати загальні вимоги до функціонального поведінки проектованої системи;

– розробити вихідну концептуальну модель системи для її подальшої деталізації у формі логічних і фізичних моделей.

Суть діаграми прецедентів полягає в наступному. Проектована система представляється у вигляді безлічі сутностей або акторів, що взаємодіють з системою за допомогою варіантів використання. При цьому актором або дійовою особою називається будь-яка сутність, що взаємодіє з системою ззовні

Діаграма прецедентів для даної системи відображена на рисунку 3.1.

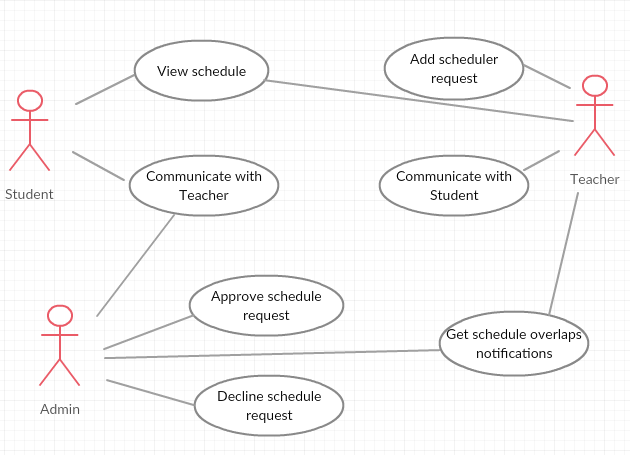


Рисунок 3.1 – Діаграма прецедентів

Діаграма прецедентів моделює систему з точки зору користувача і відображає доступні йому можливості. Діаграми цього типу допомагають при аналізі вимог до системи та проектуванні інтерфейсу користувача.

На рисунку 3.1 зображені основні функції системи, що доступні користувачам. При переході на сайт користувач потрапляє на головну сторінку, де представлена інформація про розклад роботи аудиторій, юзеру також надається можливість увійти у систему для доступу к роле-залежним функціям.

Основний функціонал, доступний користувачу системи включає в себе наступні можливості:

– реєстрація користувачів у системі;

– перегляд розкладу занять в аудиторіях учбового центру;

– інтеграція системи з розкладом CIST для відповідних аудиторії, груп та викладачів;

– надсилати запити на додання пар до розкладу аудиторії (з рядом критеріїв, таких як час проведення, пари, кількість студентів, зайнятість комп’ютерів та ін.);

– отримувати сповіщення про зміни в розкладі та про накладання пар;

– приймати/відхиляти запити на додання пар до розкладу, додавати викладачів до системи;

– надання можливостей комунікацій між студентами та викладачами;

– надання можливостей комунікацій між викладачами та адміністраторами;

– можливість отримувати інформацію про стан завантаженості аудиторії для оптимального використання ресурсів.

Фізичне представлення програмної системи разом з інформацією про те, на якій платформі вона реалізована, відображається на діаграмі розгортання.

Система для формування розкладу занять у навчальному центрі складається з таких основних частин:

– серверу (Node.js);

– веб–кліенту (chromium–based браузери);

– бази даних (PostgreSQL).

Діаграма розгортання – діаграма в UML [17], на якій відображаються обчислювальні вузли під час роботи програми, компоненти, та об'єкти, що виконуються на цих вузлах. Компоненти відповідають представленню робочих екземплярів одиниць коду. Компоненти, що не мають представлення під час роботи програми на таких діаграмах не відображаються; натомість, їх можна відобразити на діаграмах компонент. Діаграма розгортання відображає робочі екземпляри компонент. Вона застосовується для представлення загальної конфігурації і топології розподіленої програмної системи і містить розподіл компонентів по окремих вузлах системи. Крім того, діаграма розгортання показує наявність фізичних сполук-маршрутів передачі інформації між апаратними пристроями, задіяними в реалізації системи. Діаграма компонентів допомагає виявити вузькі місця системи і реконфігурувати її топологію для досягнення необхідної продуктивності.

Взаємодія компонентів системи представлена на діаграмі розгортання, що зображена на рисунку 3.2. На ній показані всі зв’язки та деякі нефункціональні вимоги, такі як: chromium–based браузери.

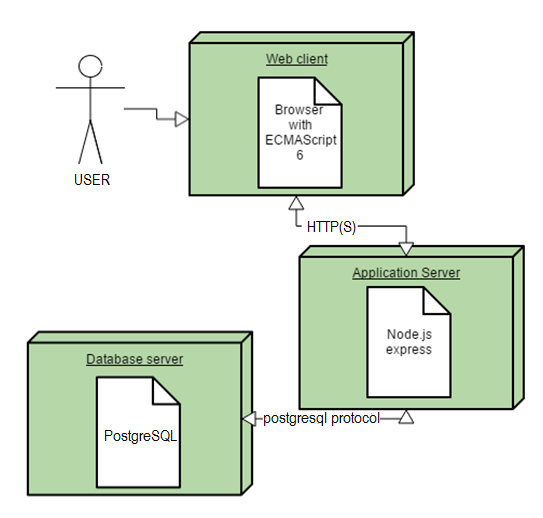


Рисунок 3.2 – Діаграма розгортання

Веб клієнт був побудований за допомогою сучасних технологій HTML5/CCS3, TypeScript, а також з використанням фреймворку Angular5 та бібліотеки AngularMaterial2 [18]. Серверна частина додатку була розроблена за допомогою Node.js з використанням фреймворку express. Node.js був обраний у зв’язку з тим, що розробка як backend частини, так і frontend ведеться однією мово програмування, та дані між частинами додатку передаються у JSON–форматі, що є нативним для JavaScript. В якості сховища даних була обрана СУБД PostgreSQL.

3.2 Аналіз та вибір технологій розробки

Веб клієнт був побудований за допомогою сучасних технологій HTML5/CCS3 [19], TypeScript, а також з використанням фреймворку Angular5 та бібліотеки AngularMaterial2. JavaScript – динамічна, об’єктно–орієнтовна мова програмування. Представляє собою стандарт ECMAScript. Найчастіше використовується як частина браузера, що надає можливість коду на стороні клієнта (такому, що виконується на пристрої кінцевого користувача) взаємодіяти з користувачем, керувати браузером, асинхронно обмінюватися даними з сервером, змінювати структуру та зовнішній вигляд веб–сторінки. JavaScript [20] класифікують як прототипну скриптову мову програмування з динамічною типізацією. TypeScript – мова програмування, представлена Microsoft восени 2012; позиціонується як засіб розробки веб-застосунків, що розширює можливості JavaScript. TypeScript є зворотньо сумісним з JavaScript. Фактично, після компіляції програму на TypeScript можна виконувати в будь-якому сучасному браузері або використовувати спільно з серверною платформою Node.js. Нововведення у TypeScript в порівнянні з JavaScript підвищують швидкість розробки, читабельність коду, рефакторинг і повторне використання коду, здійснювати пошук помилок на етапі розробки та компіляції, а також швидкодію програм.

Angular [21] це JavaScript–фреймворк з відкритим програмним кодом, який розробляє Google. Призначений для розробки односторінкових застосунків, що складаються з одної HTML сторінки з CSS і JavaScript. Його мета –розширення браузерних застосунків на основі шаблону Модель–вид–контролер (MVC), а також спрощення їх тестування та розробки. Фреймворк працює зі сторінкою HTML, що містить додаткові атрибути і пов'язує області вводу або виводу сторінки з моделлю, яка являє собою звичайні змінні JavaScript. Значення цих змінних задаються вручну або отримуються зі статичних або динамічних JSON–даних. Двостороннє зв'язування даних в Angular є найбільш примітною особливістю і зменшує кількість коду, звільняючи сервер від роботи з шаблонами. Замість цього, шаблони відображаються як звичайний HTML, наповнений даними, що містяться в області видимості, визначеній у моделі. Сервіс $scope в Angular стежить за змінами в моделі і змінює розділ HTML–вирази в уявленні через контролер. Крім того, будь–які зміни в уявленні відображаються в моделі. Це дозволяє обійти необхідність маніпулювання DOM і полегшує ініціалізацію та прототипування веб–додатків, а також підвищує продуктивність і спрощує тестування.

AngularMaterial2 був обраний як фронтенд фреймворк. Це набір готових компонентів для Angular2+ проектів, а також набір кольорових схем та стилістичних рішень для підвищення швидкості розробки та дотримання єдиного дизайнерського рішення. Цей набір інстрементів створений для полегшення розробки веб-застосунків та сайтів. Він включає CSS та HTML для типографії, форм, кнопок, таблиць, сіток, навігації тощо, а також додаткові розширення. Основними перевагами є наступне: економія часу розробки; масштабованість на різноманітні пристрої та роздільну здатність екрану без жодних змін у розмітці сторінки; усі компоненти платформи використовують єдиний стиль та шаблон за допомогою центральної бібліотеки, що дозволяє узгоджувати дизайн сторінок; простота у використанні фреймворку; сумісність з усіма популярними браузерами.

Серверна частина додатку була розроблена за допомогою Node.js з використанням фреймворку express. Node.js був обраний у зв’язку з тим, що розробка як backend частини, так і frontend ведеться однією мово програмування, та дані між частинами додатку передаються у JSON–форматі, що є нативним для JavaScript.

Node.js – платформа з відкритим сирцевим кодом для написання серверної частини веб–реалізації на JavaScript, автором якої є Раян Дал. Node.js [22] характеризується такими властивостями, як асинхронна однопотокова модель виконання запитів, неблокуючий ввід/вивід, система модулів CommonJS, JavaScript Google V8. Для управління модулями використовується пакетний менеджер npm. Node.js призначений для відокремленого виконання високопродуктивних мережевих застосунків на мові JavaScript. Функції платформи не обмежені створенням серверних скриптів для веб, платформа може використовуватися і для створення звичайних клієнтських і серверних мережевих програм. Для забезпечення обробки великої кількості паралельних запитів Node.js задіює асинхронну модель запуску коду, засновану на обробці подій в неблокуючому режимі і визначенні обробників зворотніх викликів.

В якості сховища даних була обрана СКБД PostgreSQL. PostgreSQL - об'єктно-реляційна система керування базами даних (СКБД). Порівняно з іншими проектами з відкритим кодом, PostgreSQL не контролюється якоюсь однією компанією, її розробка можлива завдяки співпраці багатьох людей та компаній, які хочуть використовувати цю СКБД та впроваджувати у неї найновіші досягнення. Сервер PostgreSQL написаний на мові C. Зазвичай розповсюджується у вигляді набору текстових файлів із сирцевим кодом. Фундаментальна характеристика об'єктно-реляційної бази даних - це підтримка об'єктів і їх поведінки, включаючи типи даних, функції, операції, домени і індекси. Це робить PostgreSQL неймовірно гнучким і надійним. Серед іншого, він вміє створювати, зберігати та видавати складні структури даних. Він забезпечує розширену ємність даних і заслужив довіру дбайливим ставленням до цілісності даних.

Додаток був розміщений на Heroku. Heroku – хмарна PaaS–платформа, що підтримує ряд мов програмування (у тому числі Node.js). Програми, які працюють на Heroku, використовують також DNS–сервер Heroku (зазвичай додатки мають доменне ім'я виду «ім’я\_додатку.herokuapp.com»). Для кожної програми виділяється кілька незалежних віртуальних процесів, які називаються «dynos». Вони розподілені за спеціальною віртуальною сіткою («dynos grid»), яка складається з декількох серверів.

Завдяки використанню адаптивного веб–дизайну, забезпечується коректне відображення веб–сторінок на різних пристроях. Метою адаптивного веб–дизайну є універсальність веб–сайту для різних пристроїв. Для того, щоб веб–сайт було зручно переглядати з пристроїв різних роздільних здатностей та форматів, за технологією адаптивного веб–дизайну не потрібно створювати окремі версії веб–сайту для окремих видів пристроїв. Один сайт може працювати на смартфоні, планшеті, ноутбуці та телевізорі з виходом в інтернет, тобто на всьому спектрі пристроїв.

У результаті була отримана архітектура додатку, що зображена на рисунку 3.3.

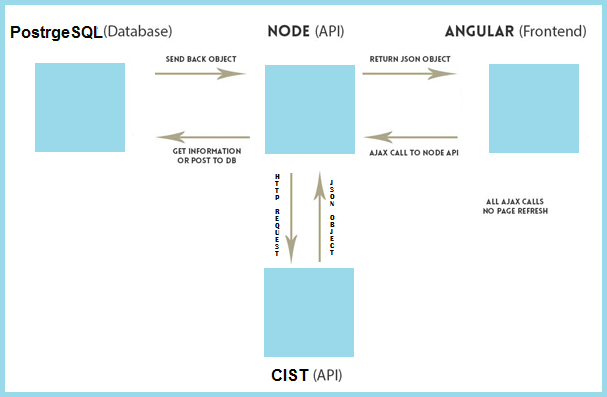


Рисунок 3.3 – Архітектура додатку

Git буде використаний для контрою версій під час розробки додатку. Git – розподілена система керування версіями файлів та спільної роботи. Git є однією з найефективніших систем керування версіями, що надає гнучкі засоби розробки.

**4 ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЯ**

Для створення інтерфейсу було використано сучасний стек технологій: HTML5 / CSS3 / JavaScript з використанням фреймворку AngularMaterial2. Цей фреймворк реалізує набір різноманітних компонентів, беручи за основу новий material дизайн від Google.

AngularMaterial2 надає набір компонентів, таких як діалогові вікна, елементи форм, елементи навігацій, компоненти для побдови інтерфейсі, кнопки та нотифікацій, таблиці. Компоненти просто інтегруються до власних Angular2+ компонентів, що врази підвищує швидкість розробки, допомагає дотримуватися єдиного дизайну, особливо враховуючи той факт, якщо у розробника немає дизайнерських навичок. Використання цих інструментів дало можливість створити зручний та зрозумілий інтерфейс користувача, що виглядає сучасно, швидко реагує на зміни та адаптується під різні браузери. Дизайн сайту був розроблений враховуючі останні тенденції UI/UX підходів.

Інтерфейс системи для формування розкладу занять у навчальному центрі створено з використанням технології RWD (респонсів веб-дизайн) [23] - дизайну веб-сторінок, що забезпечує оптимальне відображення та взаємодію сайту з користувачем незалежно від роздільної здатності та формату пристрою, з якого здійснюється перегляд сторінки. Популярність адаптивного веб-дизайну зростає з кожним днем так як вже зараз кількість мобільного трафіку сягає більше половини від всього інтернет трафіку. Ця тенденція настільки поширена, що Google 21 квітня 2015 року запустив у своїй пошуковій системі алгоритм оцінки сайту на відповідність принципам «дружного» до мобільних пристроїв інтерфейсу. Від цього показника залежить як високо сторінка буде представлена в результатах мобільного пошуку, а отже дана оцінка частково діє як штраф для сайтів, які не відповідають стандартам інтерфейсу для мобільних пристроїв.

Метою респонсів веб-дизайну є практичне відображення інформації та зручна навігація на всіх пристроях із доступом до інтернету. За технологією респонсів веб-дизайну не потрібно створювати окремі версії веб-сайту. Даний принцип базується на використанні медіа-запросів в css (каскадні таблиці стилів). На рисунку 4.1 наведено приклад використання медіа-запиту для розміру екрану пристрою до 640 пікселів.

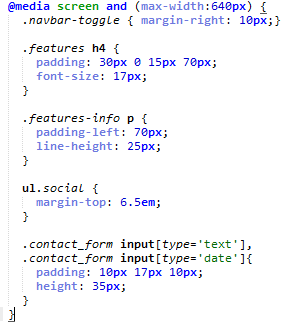


Рисунок 4.1 – Використання медіа-запросів у респосів веб-дизайні

Розроблена система буде використовуватися різними типами користувачів, а саме студентами, викладачами НВЦА та адміністратором центру. Кожному користувачеві надається різний рівень можливостей у системі. Нижче розглянемо кожного користувача окремо.

4.1 Опис функціоналу системи для користувача типу «студент»

Після відкриття сайту користувач потрапляє на головну сторінку сайту на якій представлено інформацію про поточний розклад «Навчального Науково-Виробничого Центру аутсорсингу», що діє на базі ХНУРЕ. На сторінці відображені виключно пари, що проводить навчальний центр. Що надає можливість користувачам типу «студент» швидко отримати доступ до необхідної інформації і дає повне уявлення про призначення створеного сервісу для подальших користувачів.

На рисунку 4.2. представлено інтерфейс для незареєстрованого користувача – студента.

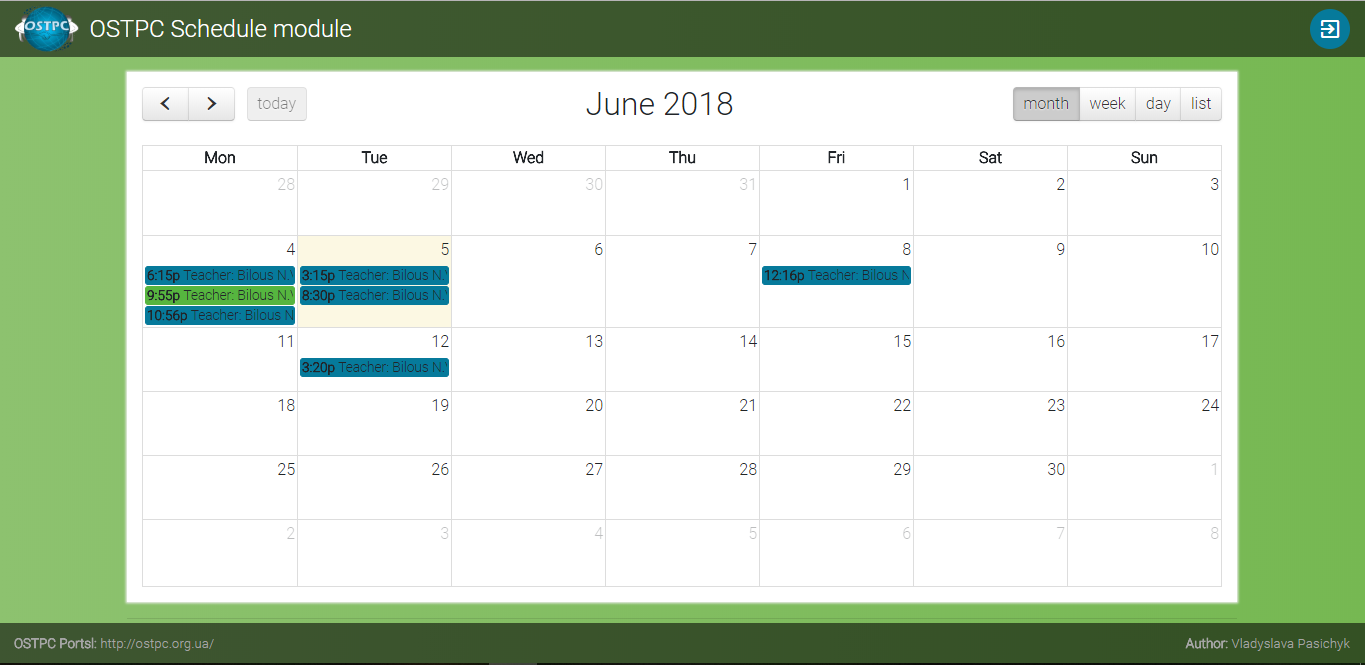


Рисунок 4.2 – Головна сторінка для користувача, що не ввійшов до системи

Сторінка реалізована в приємній кольоровій гамі спокійного зеленого відтінку. На головній сторінці представлені елементи навігацій. У шапці сторінки (хедері) відображений логотип центру OSTPC, а також кнопка для входу на сайт.

У футері (нижній навігації) додано посилання на сторінку центру та посилання для відправлення емейлу розробнику сайту. Інтерфейс системи розроблений англійською мовою, у зв’язку з тим, що мова є міжнародною, а учні різномовні.

4.2 Опис функціоналу системи для користувача типу «викладач»

Для доступу до основних можливостей додатку користувач, що зайшов на сайт повинен зареєструватися у системі, або ввести логін і пароль, якщо він вже доданий до бази користувачів. На рисунку 4.3 представлена форма для реєстрації у системі.

Для реєстрації у системі необхідно заповнити форму, що містить електронну пошту користувача, логін для входу до системи та пароль, що має відповідати усім критеріям безпеки (бути більше 8 символів, мати літери, цифри та символи). Також у разі помилки у введені даних для логіну, коли юзера з такими параметрами не знайдено у системі, користувачу буде відображено нотифікаційне повідомлення про помилку входу у систему.

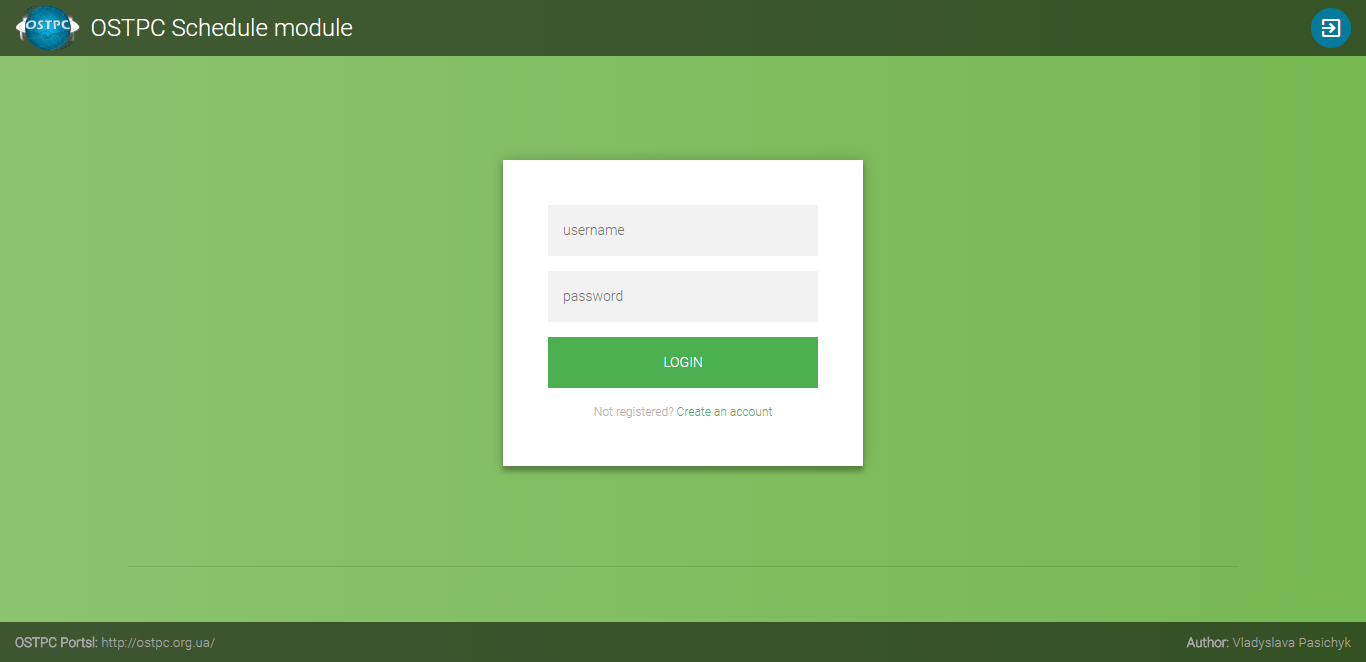


Рисунок 4.3 – Форма реєстрації у системі

Після успішного входу до системи користувачу типу «викладач» надається доступ до основних його функцій, таких як:

– перегляд розкладу занять в аудиторіях учбового центру, враховуючи заняття центру та університету;

– надсилати запити на додання пар до розкладу аудиторії з рядом різноманітних критеріїв, таких як час проведення, пари та інше;

– отримувати сповіщення про підтвердження або відхилення пар адміністратором центру.

На рисунку 4.4 представлено інтерфейс користувача типу «викладач».

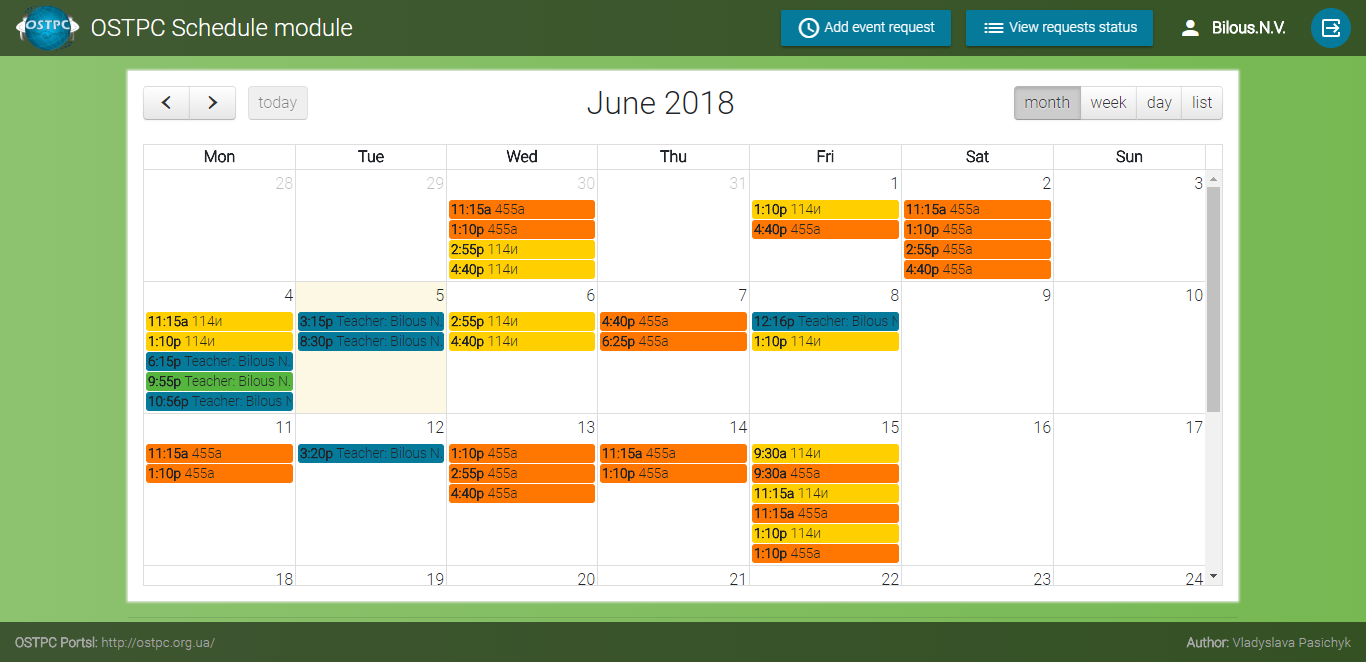


Рисунок 4.4 – Інтерфейс модуля розкладу для користувача, що зайшов до системи з роллю «викладач»

Як ми бичимо, порівнюючи рисунок 4.3 та 4.4, для викладачів у розкладі відображаються вже пари як університету так і власне центру. Для наочності ці подій мають різний колір: пари університету для аудиторій 455а та 114і відображаються жовтогарячим та жовтим кольорами відповідно. Пари центру також помічені різним кольором: для пар, що ще не були підтверджені адміністратором НВЦА було обрано синьо-голубий колір, а для тих подій, що отримали підтвердження – зелений.

Користувач типу «викладач» має можливість додавати пари до розкладу, а також переглядати статус всіх своїх запитів. Для цього у до навігації у шапці сайту було додано відповідні кнопки (див. рис. 4.5).



Рисунок 4.5 – Елементи навігації для користувача типу «викладач»

Для отримання інформацій про стан своїх запитів на додання подій до аудиторі центру, викладач повинен скористатися кнопкою «View requests status». У результаті юзеру відкриється діалогове вікно з таблице, що покаже статус усіх запитів викладача, що зараз працює з системою (див. рис. 4.6).

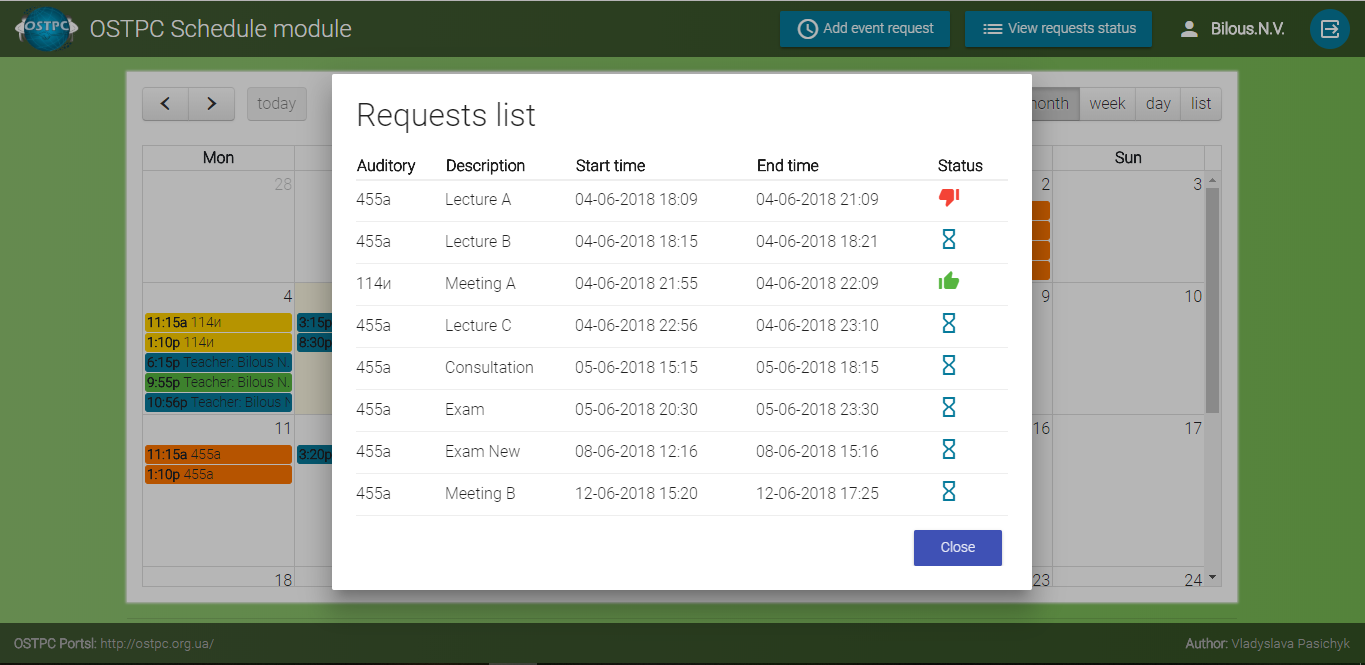


Рисунок 4.6 – Список усіх запитів викладача на додання заходів до аудиторій центру та їхній статус

У таблиці усіх запитів викладача наведено інформацію про аудиторію, в якій планується проведення заходу або навчальної пари, опис, до цього заходу, що було додано викладачем, час початку та закінчення заходу, а також статус цього запиту. Для відображення статусу заходу використовуються відповідні іконки. Таки спосіб відображання інформації було обрано, що зробити інтерфейс більш читабельним, мінімалістичним, а з іншого боку додати більше яскравості. У разі, якщо іконка, яка відображає поточний статус, буде незрозуміла певному користувачеві, чи якщо юзер забув ії значення, то при наведенні на іконку з’являється невелика інформаційна підказка у текстовому варіанті поточного статусу.

Статуси запитів відображаються трьома різними іконками. Для подій, що були підтверджені адміністратором, обрана іконка з зображенням великого пальця вгору (що символізує затвердження). Дана іконка має зелений колір, який відповідає кольору доданих до розкладу пар НВЦА. Для подій, що були відхилені адміністратором, іконка має червоний колір (колір, що привертає увагу) та зображується за допомогою великого пальця донизу (що символізує відхилення). Запити, які ще не були переглянуті адміністратором мають синій колір (як і пари в розкладі, що не є підтвердженими) та іконку пісочного годиннику.

Користувач з роллю «викладач» має можливість додавати пари до розкладу. Для цього йому потрібно викликати вікно для формування запиту на додання пари, натиснувши кнопку «Add event request» у хедері сторінки. Після цього користувачу відкриється діалог для заповнення інформації про подію, що він хоче додати.

Для додання події користувачу треба вибрати аудиторію, в якій він хоче провести подію (юзер має можливість обрати одну з двох аудиторій начального центру); визначитися, чи це буде одна подія, чи вона циклічно повторюватиметься декілька разів (вибір здійснюється за допомогою елементів керування радіо-кнопок); обрати дату та час початку та закінчення події (для вибору часу початку події був розроблений власний модуль, про який йтиметься нижче). Також користувачу надається можливість перевірити чи не збігається дана подія з розкладом групи, у разі якщо викладач хоче провести пару для студентів університетської групи. Інтерфейс даного діалогового вікна представлений на рисунку 4.7.

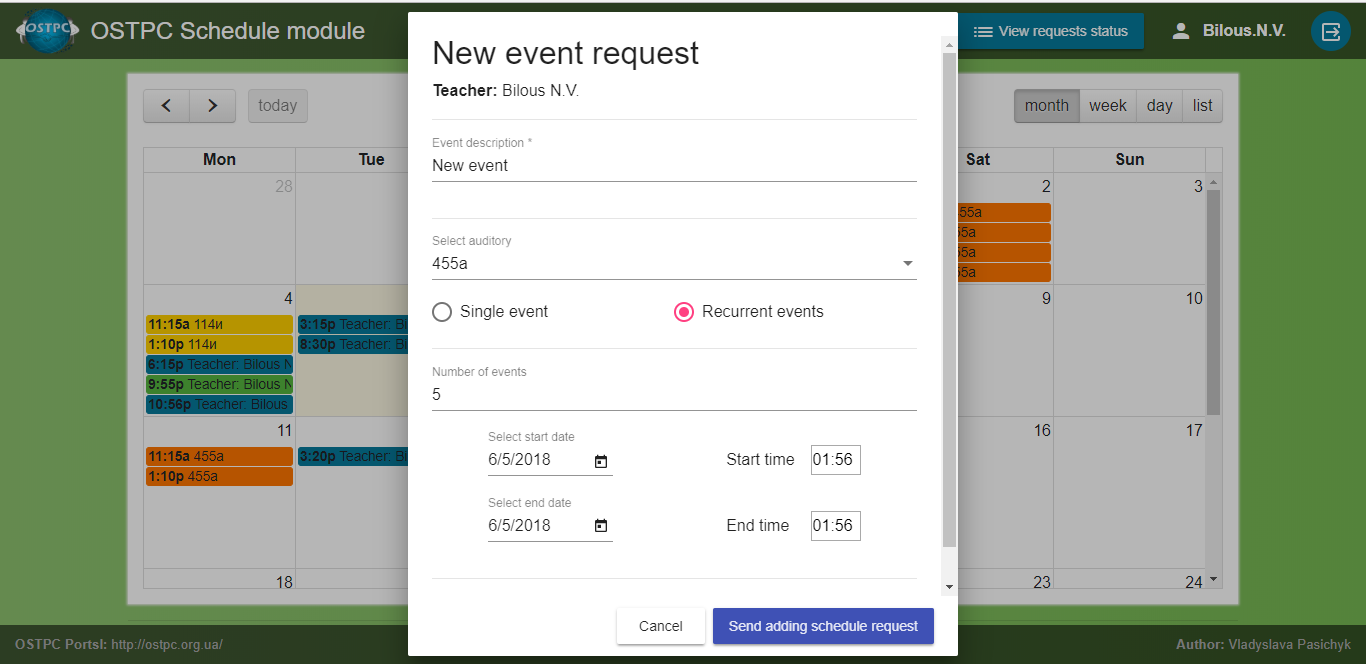


Рисунок 4.7 – Діалогове вікно для додавання запитів на проведення пари в центрі

Для вибору часу проведення заходу було реалізовано окремий модуль, що має сучасний інтерфейс у вигляді годинника, що представлений на рисунку 4.8

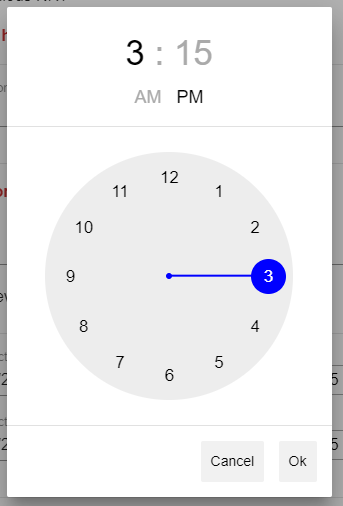


Рисунок 4.8 – Інтерфейс модулю для вибору час проведення заходу

Даний модуль дає змогу обирати час, користуючись циферблатом. Рішення є гнучким у налаштуванні та може використовуватися як сторонній модуль для інших проектів, що написані на Angular2+ версії. Інтерфейс модулю для вибору часу проведення пари представлено на рисунку 4.8. Для використання даного модулю потрібно лише додати директиву atp-time-picker до input елемента.

Після заповнення користувачем усієї необхідної інформації та спробі відправити запит на додання заходу до розкладу центру, система перевірить валідність усіх введених даних та сповістить юзера про успішне надсилання запиту (шляхом відображення спливаючого вікна внизу екрана) чи, у разі виявлення помилок, виведе інформаційні повідомлення (див. рис. 4.9).

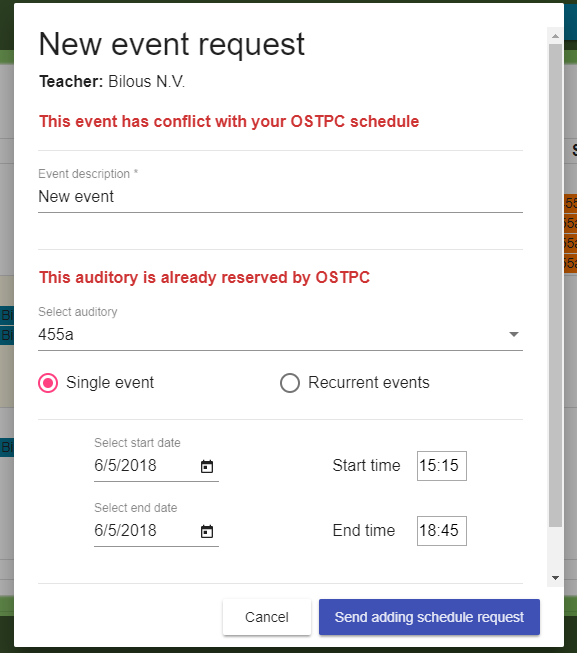


Рисунок 4.9 – Приклад повідомлення про накладання подій

Користувачу надається можливість перевірити накладання з університетським розкладом групи, у разі якщо юзер хоче провести пару для студентів університетської групи. Отримання списку факультетів та груп відбувається через відправку запиту на CIST сервер, що має відкрите API для використання. Нижче приведено код отримання даних з СIST. Було реалізоване кешування даних на клієнтській стороні, для того щоб підвищити швидкість роботи системи, зменшити кількість запитів, шляхом зберігання даних, що мають постійну основу, до локального сховища браузера користувача:

**getGroups() {**

**if (localStorage.getItem('groups')) {**

**return of(JSON.parse(localStorage.getItem('groups')));**

**}**

**return this.http**

**.get(`${CIST\_API\_URL}${GROUPS\_URL}`)**

**.pipe(**

**map((response) => {**

**const allGroups = {};**

**const allFaculties = response['university'].faculties;**

**allFaculties.forEach((faculty) => {**

**let facultyGroups = [];**

**faculty.directions.forEach((direction) => {**

**if (direction.specialities &&**

**direction.specialities.length) {**

**direction.specialities.forEach((speciality) => {**

**if (speciality.groups) {**

**facultyGroups =**

**[...facultyGroups, ...speciality.groups];**

**}**

**});**

**} else {**

**facultyGroups =**

**[...facultyGroups, ...direction.groups];**

**}**

**allGroups[faculty.id] = facultyGroups;**

**});**

**});**

**localStorage.setItem('groups',**

**JSON.stringify({allFaculties, allGroups}));**

**return {allFaculties, allGroups};**

**})**

**);**

**}**

Так як в університеті велика кількість груп і пошук по ним не є зручним, було вирішено додати крок початкового вибору факультету (використовуючи випадний список), шляхом цього зменшити кількість груп для пошуку. Було використано компонент для автокомпліту з бібліотеки AngularMaterial2 для полегшення пошуку студентської групи (див. рис. 4.10).

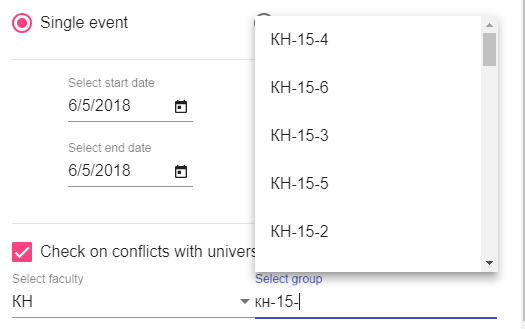


Рисунок 4.10 – Приклад використання компоненту autocomplete з бібліотеки AngularMaterial2

Перед відправленням запиту на додання пар система перевіряє ряд обмежень, а саме:

– чи коректно введені дані для часу початку та закінчення заходу (час початку повинен бути менше ніж час закінчення);

– чи вільна дана аудиторія в обраний час;

– чи не зайняти викладач в університеті у зазначений час;

– чи не зайняти викладач в центрі в обраний час;

– чи вільна обрана група, у разі якщо проводиться заняття для певної університетської групи.

У разі не валідності однієї з вищеперерахованих умов, користувачу буде показане відповідне повідомлення та надана можливість змінити параметри запиту. Після успішного надсилання запиту, користувач автоматично потрапляє на сторінку з вже оновленим розкладом центру.

4.3 Опис функціоналу системи для користувача типу «адміністратор»

У разі якщо користувач зайшов до системи використовуючи дані адміністратора, то на головному екрані буде відображено розклад аудиторій центру враховуючи як пари університету, так і пари власне НВЦА (див. рис. 4.11).

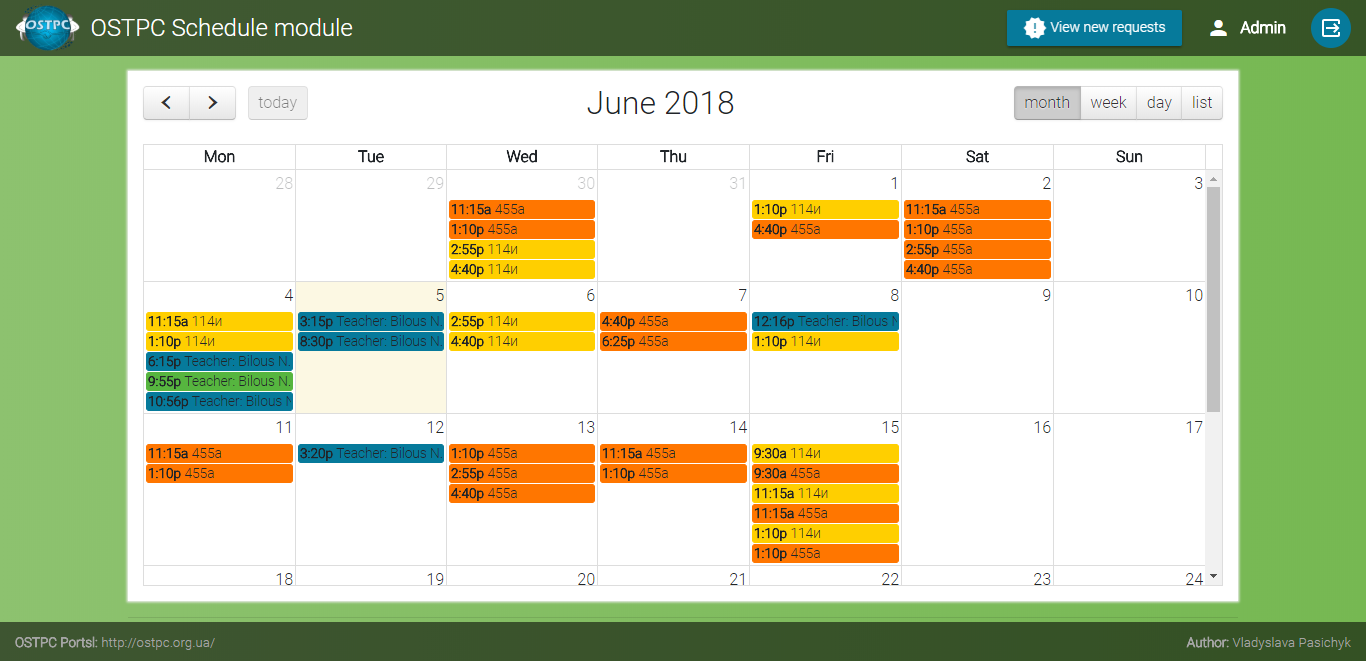


Рисунок 4.11 – Інтерфейс системи для користувача з роллю «адміністратор»

Адміністратору доступна можливість переглядати усі вхідні запити. Для отримання списку запитів треба натиснути на кнопку «View new requests». Користувачу надається можливість перевірити нові запити та змінити їхній статус. Для зміни статусу треба натиснути на іконку з підтвердженням події (зелена галочка) чи ії відхиленням (червоний хрестик). Кольори для іконок консистентні з іншими кольоровими позначеннями в системі та є інтуїтивно зрозумілими. Після відхилення події вона видалить з розкладу. У разі підтвердження заходу, він змінить свій колір у системі з синього на зелений и буде відображений в розкладі. Після зміни статусу події викладач отримає сповіщення про те, що його запит було змінено.

На рисунку 4.12 представлено форму з списком усіх нових запитів на додання подій до розкладу центру.

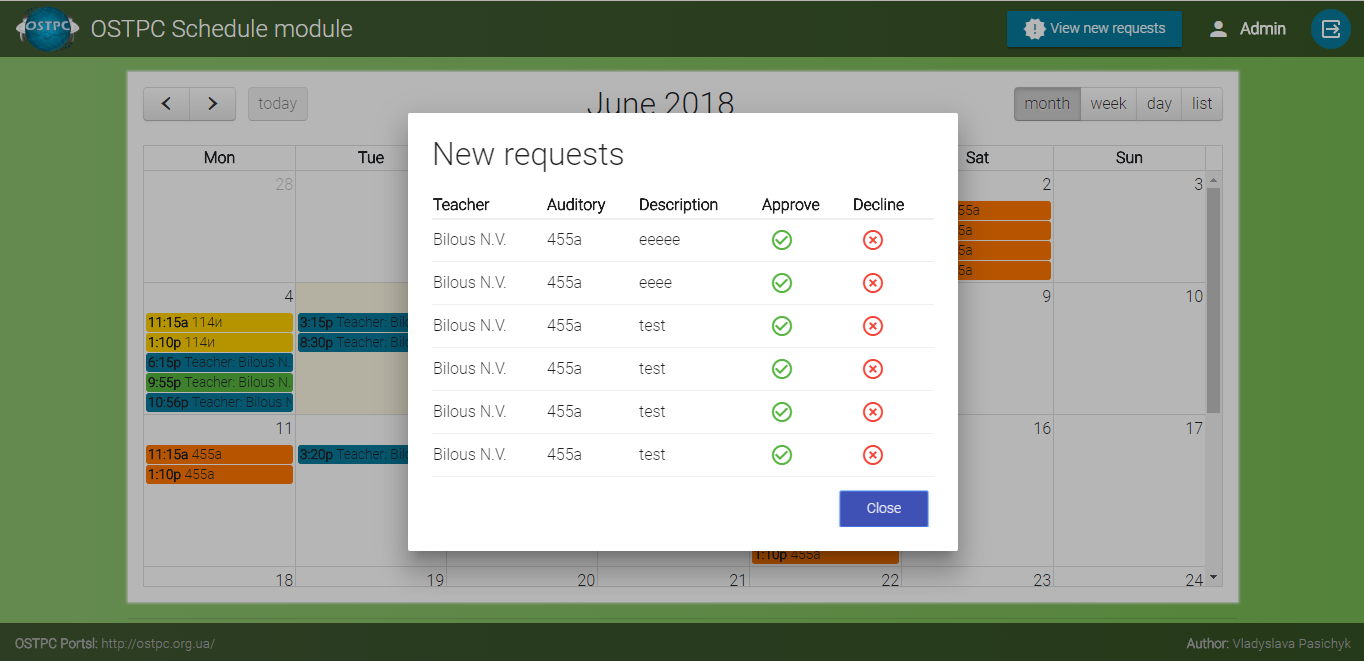


Рисунок 4.12 – Форма з новими запитами на додання подій до розкладу центру

Окрему увагу звернемо на власне календар, що відображає розклад занять в аудиторіях навчального центру. Для зручності використання користувачем модулю календаря було додано ряд елементів навігації , які зображені на рисунку 4.13. Компонент календарю має наступні навігаційні елементи (зазначено зліва направо): кнопки перемикання на попередні та наступні місяці/тижні/дні залежності від поточного типу відображення календаря; кнопка «today», що повертає до кроку, де міститься поточний день; кнопки перемикання між різними видами відображення календаря.

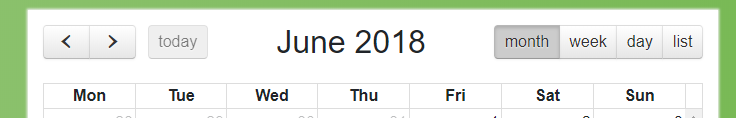


Рисунок 4.13 – Навігаційні елементи компоненту для відображення розкладу

Нижче на рисунку 4.14 представлене відображення у вигляді «список», усі події відображені один за одним та відсортовані по днях.

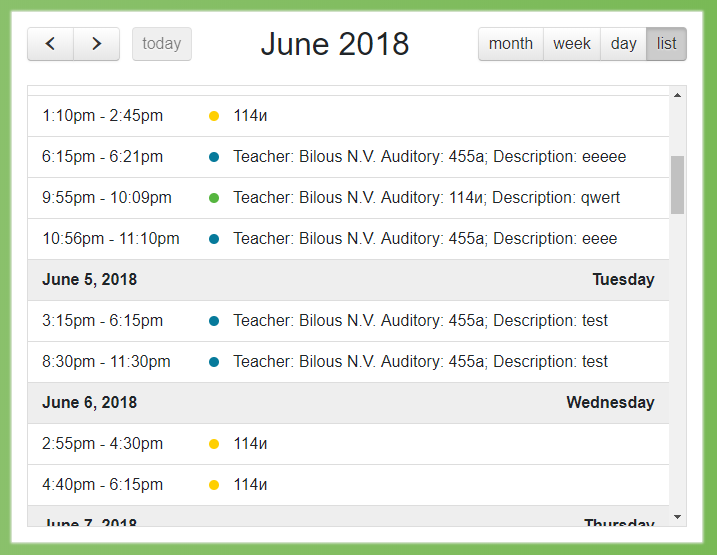


Рисунок 4.14 – Компонент «календар» в режимі відображення «список» подій

Також користувачу доступні такі режими відображення як «місяць»; «тиждень»; «день»; «список», що відображає усі події у центрі у вигляді списку, є зручним у використанні, бо детально показує опис заходів.

**5 ТЕСТУВАННЯ І ЕКСПЛУАТАЦІЯ СИСТЕМИ**

5.1 Тестування розробленої системи

У ході роботи було проведено тестування основних функцій системи.

Тестування програмного забезпечення – перевірка відповідності між реальним і очікуваним поведінкою програми, здійснювана на кінцевому наборі тестів, обраному певним чином.

Усі тести діляться на чотири групи: модульне тестування, інтеграційне тестування, тестування компонентів інтерфейсу, і системне тестування. Основні рівні в процесі розробки, як визначено в SWEBOK є модульне, інтеграційне і системне тестування, які відрізняються метою тестування, не маючи на увазі певну модель процесу.

Модульне тестування – відноситься до тестів, які перевіряють функціональність певного розділу коду, зазвичай на функціональному рівні. В об'єктно-орієнтованому середовищі, це, як правило, тестування на рівні класу, а мінімальні модульні тести містять у собі конструктори та деструктори. Модульне тестування (юніт тестування) – полягає в ізольованій перевірці кожного окремого елемента шляхом запуску тестів в штучному середовищі. Модульне тестування спрямоване на усунення помилок проектування. Ця стратегія спрямована на підвищення якості одержуваного ПЗ, до такого рівня, як вимагає процес контролю якості.

Інтеграційне тестування є типом тестування ПЗ, яке прагне перевірити інтерфейси між компонентами від програмного дизайну. Програмні компоненти можуть бути інтегровані як в рамках ітеративного підходу, так і всі разом. Інтеграційне тестування працює над виявленням дефектів у інтерфейсах та взаємодії інтегрованих компонентів (модулів).

Системне тестування – тестує інтегровану систему для перевірки відповідності всім вимогам. Системне інтеграційне тестування перевіряє, чи система інтегрується в будь-яку зовнішню систему (або системи) відповідно до системних вимог. Основним завданням системного тестування є перевірка як функціональних, так і не функціональних вимог до системи в цілому. При цьому виявляються дефекти, такі як невірне використання ресурсів системи, непередбачені комбінації даних користувача рівня, несумісність з оточенням, непередбачені сценарії використання, відсутня або невірна функціональність, незручність використання.

Розроблена система має підтримувати наступні основні функції:

– реєстрація користувача у системі;

– отримання розкладу занять користувачами;

– надання можливостей додання пар до розкладу центру користувачем типу «викладач» , використовуючи критерії аудиторій, дати та часу початку та закінчення пари, вводу опису події, перевірки на накладання з розкладом груп студентів університету;

– надання можливості викладачу переглядати статус усіх своїх запитів на додання подій до календаря НВЦА;

– надання можливості адміністратору керувати заходами, шляхом можливості їхнього підтвердження чи відхилення.

Було проведене як автоматизоване [24], так і ручне тестування системи. Для написання тестів використовувався фреймворк Jasmine. Вручну було протестовано робочі області програмної системи.

У ході роботи усі розроблені Angalur компоненти та сервіси біло покрито unit тестами, для написання яких використовувався фреймворк Jasmine. Jasmine це вільний фреймворк для тестування коду написаного мовою JavaScript. Його може бути запущено на будь-якій платформі, де виконується JavaScript. Він не втручається в роботу ні програм, ні IDE, а також має дуже простий для читання інтерфейс. До основних переваг даного фреймворку можна віднести підтримку асинхронного тестування; можливість використання "шпигунів" для впровадження тестових «дублерів»; підтримку тестування front-end коду через front-end розширення Jasmine під назвою Jasmine-jQuery. Jasmine забезпечує багатий набір вбудованих функцій для порівнянь, які повертають логічне значення: true, якщо сподівання вірне (спосіб показати, що тест пройшов) або false, якщо очікування не збігається. Хороша практика полягає в тому, щоб використовувати єдине очікування в it() специфікації тесту.

Jasmine включає в себе такі вбудовані функцій для порівнянь: toBe, toBeTruthy, toBeFalsy, toContain, toBeDefined, toBeUndefined, toBeNull, toBeNaN, toBeGreaterThan, toBeLessThan, toBeCloseTo. toBe перевіряє, чи є обидва речі одним і тим же об'єктом. Функції toBeTruthy, toBeFalsy чи є очікування true чи false, а функція toBeDefined, toBeUndefined чи є очікування визначеним чи ні. Функція toContain використовується для перевірки того, що елемент, об'єкт або підрядок містяться в масиві, списку або рядку. Jasmine має також багато інших можливостей, таких як пошук за фільтром («custom matchers»), «шпигуни» (spies) та асинхронні специфікації («asynchronous specifications»).

Нижче приведено простий тестовий пакет, для перевірки ініціалізації app компоненту:

**import { TestBed, async } from '@angular/core/testing';**

**import { AppComponent } from './app.component';**

**describe('AppComponent', () => {**

**beforeEach(async(() => {**

**TestBed.configureTestingModule({**

**declarations: [**

**AppComponent**

**],**

**}).compileComponents();**

**}));**

**it('should create the app', async(() => {**

**const fixture = TestBed.createComponent(AppComponent);**

**const app = fixture.debugElement.componentInstance;**

**expect(app).toBeTruthy();**

**}));**

**it(`should have as title 'app'`, async(() => {**

**const fixture = TestBed.createComponent(AppComponent);**

**const app = fixture.debugElement.componentInstance;**

**expect(app.title).toEqual('app');**

**}));**

**it('should render title in a h1 tag', async(() => {**

**const fixture = TestBed.createComponent(AppComponent);**

**fixture.detectChanges();**

**const compiled = fixture.debugElement.nativeElement;**

**expect(compiled.querySelector('h1').textContent).toContain('Welcome to app!');**

Для запуска тестів використовувався test-runner Karma. Це утиліта для тестування, основними перевагами якої є:

– можливість запускати тести з консолі;

– можливість автоматично проганяти усі тести при кожному зберіганні змін у коді;

– можливість писати тести на багатьох фреймворках, таких як jasmine, qunit та ін.;

– можливість проганяти тести вдразу на декількох браузерах (у тому числі віртуальних, наприклад fantomjs).

Karma звісно можна використовувати не тільки для тестування angular, але і в цілому для будь-якого іншого клієнтського коду на js. Karma легка та зручна у використанні.

Для зменшення ручного тестування було проведене наскрізне тестування. Наскрізне тестування - це поведінкове тестування за методом "чорного ящика" (перевірка виконання додатком заданих функціональних вимог, при якому не використовуються знання про внутрішню структуру об'єкта що тестується). Тобто тестується, що система працює, як планувалося, з точки зору кінцевого користувача, якому все одно, чи працює додаток «як планується», йому важливо, щоб функціонал працював у відповідності зі його власними очікуваннями. Тестування по суті це автоматизований запуск додатку у браузері та послідовні дії, що перевіряють весь функціонал користувацького інтерфейсу. Виконувати усі ці дії вручну є дуже неефективно, тому були створені e2e тести.

Protractor – фреймворк, що був використаний для наскрізного тестування. На відміну від стандартного виконувача сценаріїв Angular, Protractor розроблений на основі Selenium WebDriver - інструменту для автоматизованого тестування веб-додатків, з API та набором розширень, що дозволяють керувати поведінкою браузера. Розширення WebDriver є для всіх типів браузерів, включаючи найбільш популярні. Таким чином, ми можемо швидко та стабільно тестувати в реальному браузерному середовищі. Protractor працює в зв'язці з Jasmine. Protractor можна встановити як самостійний виконавець тестів або використовувати як бібліотеку.

Нижче приведено приклад тестового пакету для головного модуля системи:

**const appPage = new AppPage();**

**describe('Application page not logged in :', () => {**

**beforeAll(() => {**

**appPage.navigateViaPathTo();**

**browser.sleep(BaseE2E.delay);**

**});**

**it('Check elements existing', () => {**

**Object**

**.keys(AppPage.appElements)**

**.filter((keyElement: string) =>**

**keyElement !== 'homeButton' &&**

**!keyElement.match('ButtonMenuItem'))**

**.forEach((keyElement: string) => {**

**expect(AppPage.appElements[keyElement].element.isPresent())**

**.toBeTruthy(TestErrorMessagesService.getPresentErrorMessage(**

**AppPage.appElements[keyElement].title));**

**});**

**Object**

**.keys(AppPage.appInputElements)**

**.filter((keyInputElement: string) =>**

**!keyInputElement.match('ButtonMenuItem'))**

**.forEach((keyInputElement: string) => {**

**expect(AppPage.appInputElements[keyInputElement].element.isPresent())**

**.toBeTruthy(TestErrorMessagesService.getPresentErrorMessage(**

**AppPage.appInputElements[keyInputElement].title));**

**});**

**Object**

**.keys(AppPage.appSelectElements)**

**.filter((keySelectElement: string) =>**

**!keySelectElement.match('ButtonMenuItem'))**

**.forEach((keySelectElement: string) => {**

**expect(AppPage.appSelectElements[keySelectElement].element.isPresent())**

**.toBeTruthy(TestErrorMessagesService.getPresentErrorMessage(**

**AppPage.appSelectElements[keySelectElement].title));**

**});**

У коді приведеному вище продемонстроване перевірка наявності елементів на сторінці, коли користувач не ввійшов до системи

Ручне тестування [25] – це частина процесу тестування на етапі контролю якості в процесі розробки програмного забезпечення. Воно проводиться тестувальником без використання програмних засобів, для перевірки програми або сайту шляхом моделювання дій користувача. У ролі тестувальників можуть виступати і звичайні користувачі, повідомляючи розробникам про знайдені помилки.

Тестування було проведено за допомогою динамічного та статичного підходів. Динамічний підхід включає в себе запуск програмної системи, статичний підхід перевірку синтаксису коду програми.

У ході тестування було виявлено, що система працює коректно та відповідно до постановки задачі. Користувач може зареєструється у системі. Розклад викладача автоматично завантажується. Додані нові таски правильно зберігаються та відображаються. Статуси запитів є актуальними і відображаються коректно. Адміністратор має змогу змінити статус запитів, що точно вносить зміни до поточного розкладу.

Таким чином, у ході проведеного тестування було встановлено, що всі основні функції системи працюють відповідно.

5.2 Експлуатація системи

У ході атестаційної роботи магістра було спроектовано та розроблено програмну систему у вигляді веб- додатку, яка має стати зручним інструментом для підтримки формування розкладу у Навчальному Науково-Виробничому Центрі аутсорсингу, що працює на базі університету ХНУРЕ. Система надає можливість користувачу типу «студент» переглядати розклад центру; користувачу типу «викладач» надається можливість відправляти запити на додання подій до календаря НВЦА, отримувати сповіщення про зміни в їх статусі та у розклад. Користувач-адміністратор керує запитами на додання пар, сповіщає викладачів про зміни в розкладі. Інтерфейс програмної системи дозволяє зручно переглядати розклад центру, гнучко вносити зміни до нього, додаючи різні додаткові пари та заходи. Важливою особливістю системи є інтеграція з зовнішнім сервісом CIST – розкладом університету, що дозволяє враховувати зайнятість аудиторій, викладача, групи в учбовому процесі університету під час додання запитів на проведення заходу у навчальному центрі.

Програмна система призначена для формування розкладу у навчальному центрі НВЦА, що діє на базі університету ХНУРЕ, що дозволить автоматизувати та полегшити процес, що є зараз. Вона буде корисна користувачам тим, що система надає можливість порівнювати велику кількість учбових розкладів з розкладом власне центру для пошуку вільних годин для проведення заходів в навчальному центрі. Маючи сучасний, простий та зрозумілий інтерфейс, система є проста у користуванні та дозволяє значно зменшити час виконання вищезазначених задач, завдяки частковій автоматизації.

У результаті проведеного тестування було виявлено декілька недоліків у інтерфейсі програмного продукту, а саме занадто яскраву кольорову схему, відсутність легенди кольорових позначень, а також загального опису системи – головної сторінки. Отже, в подальших кроках покращення програмної системи буде розроблено та застосовано більш легкий та мінімалістичний інтерфейс модулю розкладу та системи загалом, усунуті вищеперераховані зауваження.

Розроблена система орієнтована на використання викладачами та студентами НВЦА, а також його адміністратором. Додаток є актуальним у зв’язку з відсутністю на ринку подібних програм для вирішення вищезазначених задач. Система надає можливість ефективно керувати розкладом навчального центру, що діє на базі університету.

**ВИСНОВКИ**

У ході атестаційної роботи магістра було проведено дослідження сучасних підходів формування розкладу а можливостей CRM для роботи з інтеграцією розкладів зі сторонніх сервісів, проведено аналіз діяльності НВЦА, спроектована та розроблене програмне забезпечення для формування розкладу занять у навчальному центрі, якому необхідно базуючись на розкладі пар у певних аудиторіях університету оптимально розподілити ресурси навчальних лабораторій.

На початку роботи було проведено аналіз предметної області, в процесі якого були визначені основні проблеми, що існують в розглянутій галузі. Було проведено аналіз існуючих CRM систем, які можна застосовувати на навчальному центрі. Особливу увагу було приділено модулю розкладу. Було виявлено їх основні можливості та недоліки. На підставі аналізу предметної області була проведена постановка завдання.

У ході виконання поставленої задачі було:

* проведено аналіз та UML-моделювання предметної галузі;
* проведено аналіз можливостей існуючих систем;
* проведено дослідження підходів формування розкладу та розроблений алгоритм додавання заходів до розкладу центру;
* виконано програмну реалізацію веб-системи та мобільного додатку;
* проведено тестування розробленого програмного продукту.

Кінцевий продукт виконує функції, які були заплановані під час постановки завдання:

– реєстрація користувача у системі;

– отримання розкладу занять користувачами;

– надання можливостей додання пар до розкладу центру користувачем типу «викладач» , використовуючи критерії аудиторій, дати та часу початку та закінчення пари, вводу опису події, перевірки на накладання з розкладом груп студентів університету;

– надання можливості викладачу переглядати статус усіх своїх запитів на додання подій до календаря НВЦА;

– надання можливості адміністратору керувати заходами, шляхом можливості їхнього підтвердження чи відхилення;

– отримання звітів адміністратором.

Створений у процесі виконання роботи програмний продукт є актуальним для викладачів та студентів Навчального Науково-Виробничий Центр аутсорсингу, що діє на базі університету ХНУРЕ.

У перспективі розглядається вдосконалення даного продукту шляхом додавання нових можливостей таких, вдосконалення алгоритму додання заходів до розкладу навчального центру, впровадження інтеграції з персональним календарем викладача (наприклад з Google календарем) для запобігання накладань у заходах центр, університету та власних подіях.

**ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ**

1. Аллен Д. Как привести дела в порядок. Искусство продуктивности без стресса. М.: Манн, Иванов и Фербер, 2011. – 368 с.

2. НВЦА [Електронний ресурс] / Сайт НВЦА. - Режим доступу: : www/ URL : https://ostpc.org.ua/ – Загол. з екрану.

3. Синило, Л. Сложнее, чем кажется – внедрение CRM. М.: Новый Маркетинг, 2006. – 500с.

4. Prosperworks [Електронний ресурс] / Сайт сервісу Рrosperworks. - Режим доступу: : www/ URL : https://www.prosperworks.com/. – Загол. з екрану.

5. ZoroCRM – CRM that helps you sell [Електронний ресурс] / Сайт сервісу ZoroCRM. - Режим доступу: : www/ URL : https://www.zoho.eu/. – Загол. з екрану.

6. AmoCRM – Не тереяйте клиентов [Електронний ресурс] / Сайт сервісу – Режим доступу: : www/ URL : https://www.amocrm.ru/. – Загол. з екрану.

7. Бізнес працює в Bitrix24 [Електронний ресурс] / Сайт сервісу Bitrix24. - Режим доступу: : www/ URL : https://www.bitrix24.ua/. – Загол. з екрану.

8. Base - Enhance your sales team's processes, productivity and pipeline visibility [Електронний ресурс] / Сайт сервісу – Режим доступу: : www/ URL : https://getbase.com/ – Загол. з екрану.

9. Salesforce Sales Cloud - Blaze your trail with the world’s #1 platform for sales [Електронний ресурс] / Сайт сервісу – Режим доступу: : www/ URL : https://www.salesforce.com/products/sales-cloud/overview/ – Загол. з екрану.

10. Розклад ХНУРЕ СIST [Електронний ресурс] / Сайт сервісу СIST. - Режим доступу: : www/ URL : https:// cist.nure.ua/. – Загол. з екрану.

11. Фаулер, М., Дейвид, Р. Архитектура корпоративных программных приложений. М.: Вильямс, 2008. – 544 с.

12. Уорсли, Дж., Дрейк, Дж. PostgreSQL. Для профессионалов. СПб: Питер, 2003. – 496 с.

13. Пайкерс, В. Г. Методика составления расписания в образовательном учреждении. Изд. 3-е испр. и доп.. М.: АРКТИ. 2011.

14. Marx, Daniel. "Graph Coloring Problems and Their Applications in Scheduling". New Riders. 2004. – 202 с

15. Fraser, Alex (1957). «Simulation of genetic systems by automatic digital computers. I. Introduction». Aust. J. Biol. Sci. 1957. – 484–491c.

16. Фаулер, М. UML. Основи, 3-є видання: пер. з англ. СПб: Символ-Плюс., 2004. – 192 с. – ISBN 5-93286-060-Х.

17. Діаграма варіантів використання (use case diagram) [Електронний ресурс] / UML Теорія – Режим доступу: http://www. methodology.com/ . – Загол. з екрану.

18. Colborne, G. Simple and Usable Web, Mobile, and Interaction Design (Voices That Matter) 1st Edition. New Riders, 2010. – 208 c.

19. Duckett, J. HTML and CSS: Design and Build Websites. 1st Edition. John Wiley & Sons, 2011. – 490 c. – ISBN-13: 978-1118008188.

20. Флэнаган, Д. JavaScript. Подробное руководство. СПб. Символ-Плюс, 2013. – 1080 с. ISBN 978-5-93286-215-5.

21. Wahling, D. Angular in 60 Minutes. Wahlin Consulting, 2014. – 102c.

22. Документація Nodejs [Електронний ресурс] / Офіційний сайт проекту Nodejs. – Режим доступу: www/ URL: https://wwwnodejs.org/ . – Загол. з екрану.

23. Marcotte, E. Responsive Web Design. A Book Apart, 2011. – 143 с. – ISBN 978-0-9844425-7-7.

24. Винниченко, И. Автоматизация процессов тестирования. СПБ.: Питер, 2005. – 203 с.

25. Black, R. Managing the Testing Process: Practical Tools and Techniques for Managing Hardware and Software Testing 3rd Edition. Woley, 2009. – 676 c.

**ДОДАТОК А**

Публікація «Деякі проблеми застосування СRM систем для автоматизації діяльності Навчального Центру Аутсорсингу університету»

Деякі проблеми застосування СRM систем для автоматизації діяльності Навчального Центру Аутсорсингу університету

ст. гр. ІПЗм-16-1 Пасічник В.Е., ХНУРЕ, Харків, Україна

к.т.н., проф. Білоус Н.В, ХНУРЕ, Харків, Україна

Більшість людей все частіше помічають той факт, що завантаженість особистими справами та робочими завданнями настільки велика, що вони просто не знають, як встигнути виконати усі задачі. Для запобігання таких проблем існують багато методик тайм-менеджменту, що дозволяють навчитися грамотно планувати свій час. Одною із простих методик управління часом [1] є складання списку завдань з певними часовими межами. Це дозволяє збільшити ефективність використання як особистого, так і робочого часу. Але часто буває так, що для організації якогось заходу треба узгоджувати розклади великої кількості людей. Доволі складно обробити велику кількість інформації самотужки, враховуючи те, що дані динамічні та можуть змінюватися, чи, наприклад, те що треба організовувати декілька подій.

У ході роботи була проаналізована діяльність «Навчального наукового-виробничого центру аутсорсингу» [2] (далі НВЦА), що діє на базі Харківського національного університету радіоелектроніки. Основними напрямками діяльності НВЦА є проведення різноманітних курсів, тренінгів, семінарів, лекції як викладачами, так і розробниками програмного забезпечення з IT-компаній. На базі OSTPC (НВЦА) з одного боку проводяться заняття основного учбового процесу університету, а з другого – заходи власне центру, що заходяться поза межами основного навчального розкладу. Дослідивши організацію робочого процесу у навчальному центрі (а саме «Навчального науково-виробничого центру аутсорсингу») та викладачів в університеті, було відзначено, що процес створення робочого розкладу не є повністю автоматизованим. Маючи за основу розклад університетських занять в аудиторіях навчального центру, розклад занять самого центру, власний розклад, розклад студентів університету та НВЦА, викладачу доводиться зіставляти усі вищеперераховані дані, іноді навіть використовуючи їх паперові версії у зв’язку з відсутність електронних. Така робота займає багато часу, потребує уваги, спричиняє безліч незручностей та навіть може призвести до технічних помилок. Її не зручно виконувати вручну, бо доводиться постійно тримати в голові багато факторів, що впливають на розклади в різних учбових аудиторіях центру та різних навчальних групах. Отже, ця задача є актуальною, бо на даний час проблема присутня у навчальному центрі, та потребує пошуку рішень для її розв’язання.

Для оптимізації процесу складання розкладу в навчальному центрі та керування завантаженістю в аудиторіях було вирішено розробити програмне забезпечення, що автоматизує процес зіставлення розкладів з різних джерел. Враховуючи специфіку роботи навчального центру, була розглянута можливість впровадження готових рішень до використовуваних зараз у центрі програмних систем. Увага була приділена наявним на ринку CRM системам.

У даний час на ринку існує велика кількість СRM систем, які можна використовувати для керування процесами у навчальному центрі. CRM система (Customer Ralationship Management) [3] – це прикладне програмне забезпечення для організацій, призначене для автоматизації стратегій взаємодії з замовниками (клієнтами), зокрема, для підвищення рівня продажів, оптимізації маркетингу і поліпшення обслуговування клієнтів шляхом збереження інформації про клієнтів та історії взаємин з ними, встановлення і поліпшення бізнес-процесів, подальшого аналізу результатів.

1

Вибір CRM-системи є важкою задачею, зважаючи на їхню різноманітність на ринку. Під час вибору CRM системи треба враховувати усі поточні потреби бізнесу. Детально розглянувши ринок, було виявлено, що найбільш популярними на даний час CRM системами є нижче проаналізовані додатки. Слід зазначити, що особлива увага буде приділятися можливості інтегрування модулю розкладу в CRM системі з розкладами з сторонніх сервісів, так к основною задачею для навчального центру на базі університету є можливість впровадити розклад ВУЗу до свого власного.

Prosperworks [4] – CRM система, що була створена для Google, використовується та рекомендована ним. Основними її можливостями є:

- легке інтегрування з Google таблицями, документами, презентаціями, календарем, поштою, тому ії використання просте, бо з Google інструментами знайомий майже кожен і всі дані зберігаються в одному місці;

- легкий доступ до списку контактів та інтеграція з різноманітними популярними сервісами для імпортування контактних даних;

- отримання інформативних звітів, виявлення критичних проблем бізнесу, які підвищать ефективність команди та оптимізуватимуть потік маркетингу;

- висок надійність та збереження даних.

Вище перераховані переваги Prosperworks роблять дану CRM систему досить привабливою для впровадження в роботу навчального центру «Навчально-виробничого центру аутсорсингу». Але як ми бачимо, у системі має можливостей для розширення та інтегрування з розкладами зі сторонніх сервісів. Також слід звернути увагу на те, що сервіс є платним та розрахований на більш великі команди.

ZohoCRM [5] - супермен в світі управління взаємовідносинами з клієнтами на 3-х ключових рівнях: маркетинг, продажі, support. Сервіс дозволяє контролювати і автоматизувати основні бізнес-процеси. Функціонал: кроссплатформеність; Інтеграція з Facebook, Twitter, Gmail, WordPress; перевірка відвідувачів сайту і сегментація; відстеження джерел трафіку; історія взаємодій, звіти, повідомлення; управління клієнтською базою; клієнтський сервіс; зручний імпорт / експорт даних; зручний розклад завдань, календар; планування, прогноз, аналіз воронки. ZohoCRM дозволяє зібрати всі важливі дані в одному місці: контакти покупців, їх історію, інформацію про джерела трафіку, кількості продажів і т.д. Сервіс є платним.

Якщо розглядати україномовні та безкоштовні системи, то слід звернути увагу на Бітрікс24 [6] - зручна, багатофункціональна, яка вже кілька років не здає позиції в рейтингу CRM-систем. Основний упор у сервісі зроблений саме на управління проектами, завданнями, документацією, звітами, планами. Разом з цим Бітрікс24 непогано справляється з оптимізацією відносин з клієнтами. На відміну від попередніх 2-х систем, Бітрікс розрахований на впровадження у всю компанію. Основні функції: ведення та управління базою даних клієнтів; фіксація різних взаємодій (дзвінки, листи); робота з документами та фінансами; відстеження замовлень, оплат; кроссплатформеність; внутрішній чат, розклад, календар, постановник завдань, стрічка змін по проектам, база знань, бейджи; управління проектами; email-трекер - збереження історії листування всередині системи; інтеграція з соцмережами; технічна підтримка. Цей сервіс оптимізує взаємодію між різними відділами, автоматизує робочі. В огляді CRM-систем для малого бізнесу, а також для малобюджетних стартапів, Бітрікс24 - безперечний лідер як мінімум по одній важливій причині - сервісом можна користуватися безкоштовно, якщо в ньому задіяно не більше 12 осіб.

2

Згідно специфіки роботи навчального центру, що з одного боку займається навчальною діяльністю на базі університету, а з іншого діяльність за межами ВУЗу (проведення курсів, трененгів, семінарів, пошук студентів, рекламна діяльність і т.і.), особливу уваг при виборі CRM системи слід звернути на CSS (Customer Service & Support) модуль. При аналізі даного модулю у різних вищеперерахованих та інших присутніх на ринку CRM системах, особливу вагу було приділено задачі складання розкладу. Майже усі системи дають таку функцію. Але не було знайдено можливості інтеграції за сторонніми сервісами для розширення цієї саме задачі.

Навчальний центр OSTPC знаходиться на базі ХНУРЕ. Тому його учбові аудиторій частково задіяні для учбового процесу та поза навчальних заходів університету. Отже постає питання про додання пар центру до розклад. Розклад ХНУРЕ доступний до перегляду у системі CIST [7]. Даний портал надає доступ до розкладу університету, електронного журналу, а також має відкрите API для можливості інтегрування з сервісом. Простим рішенням було б додавати пари центру до сітки розкладу CIST. Але це рішення не є прийнятним у зв’язку зі специфіко роботи центру. По-перше, центр має гнучкий графік пар, а миттєве додання через навчальний центр не можливе. По-друге, сайт розкладу університету не надає можливості вибрати англійську мову, що робить пошук пар для неукраїномовних студентів центру – важким. Інтерфейс СIST також не відповідає сучасним вимогам та не заохочує до використання. По-третє, на сайті СIST на додаються розклади не учбових заходів, які також необхідно враховувати при плануванні розкладу OSTPC.

Отже, враховуючи специфіку потреб навчального центру OSTPC інтегруватися та гнучко розширювати систему розкладу ХРУРЕ CIST та відсутність таких можливостей у готових рішеннях, постає питання про реалізацію власного програмного продукту.

Складання розкладу - одна з найбільш поширених завдань в плануванні і оптимізації навчального процесу в начальних закладах. Від того, наскільки добре складений розклад, залежить ефективність роботи викладача, засвоєння навчального матеріалу студентами, раціональне використання ресурсів навчального закладу. Автоматизація складання розкладу - досить класична задача в системах управління навчальним процесом, але на даний момент немає єдиного, загальноприйнятого способу її вирішення.

3

У загальній постановці завдання складання розкладу є процес розподілу деякого кінцевого набору подій у часі в умовах ресурсних та інших обмежень. В нашій задачі в якості вихідних даних для складання розкладу виступають зайнятість викладачів у ВУЗі, у навчальному центрі, зайнятість аудиторій як університетом так і навчальним центром, завантаженість аудиторій, кількості навчальних пар.

Складання розкладу відноситься до завдань цілочисельного програмування, складність решення яких зростає експоненціально з ростом числа і можливих значень варійованих змінних (такі завдання входять до класу NP-складниз завдань). Існують підходи складання розкладу, засновані на так званих точних (класичних) методах і алгоритмах цілочисленного програмування, на методах розмальовування графів. Крім того, застосовуються методи повного перебору, гілок і меж, а також евристичні методи, в тому числі засновані на генетичних алгоритмах.

За допомогою даних методів можливе отримання точної математичної моделі, що відповідає всім обмеженням, але в силу NP-складного характеру завдання складання розкладу ця модель буде громіздкою і складною. Щоб уникнути цього, застосовується імітаційне моделювання. У цьому випадку алгоритм оперує безпосередньо розкладом та списком занять, які необхідно включити в розклад.

Отже, було запропоновано створити програмну систему, що допоможе навчальному центру OSTPC інтегруватися та розширювати систему розкладу ХРУРЕ CIST для подальшого складання та оптимізації власного учбового розкладу. Основні функції системи можна відобразити на рисунку 1.

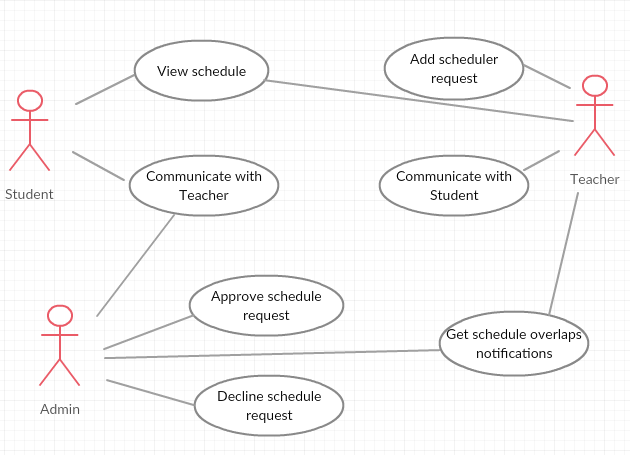


Рисунок 1 – Діаграма прецедентів до системи для навчального центру

Основний функціонал, доступний користувачу системи буде включати в себе наступні можливості:

– перегляд розкладу занять в аудиторіях учбового центру;

– інтеграція системи з розкладом CIST для аудиторії, груп та викладачів;

4

– надсилати запити на додання пар до розкладу аудиторії (з рядом критеріїв, таких як час проведення, кількість студентів, зайнятість комп’ютерів та ін.), а також перевірка на перетин розкладів викладач-аудиторія-групи;

– отримувати сповіщення про зміни в розкладі та про накладання пар;

– приймати/відхиляти запити на додання пар до розкладу, додавати викладачів до системи;

– надання можливостей комунікацій між студентами та викладачами, між викладачами та адміністраторами;

– можливість отримувати інформацію про стан завантаженості аудиторії для оптимального використання ресурсів.

До позитивних рис даного рішення слід віднести те, що розробити інформаційну систему, що реалізує усі потреби, центр може своїми силами, в достатньо короткий час та з мінімальними затратами. Також варто відмітити, що для OSTPC, як аутсорсингового центру, добре мати систему, що можна представляти як власний продукт навіть за межами університету.

Отже, в результаті аналізу діяльності та потреб НВЦА було проведено дослідження ринку для пошуку існуючих рішень для задачі автоматизації процесу складання розкладу у навчальному центрі. Враховуючи специфіку потреб OSTPC як центру, що діє на базі університету, було виявлено, що готові рішення не можуть бути використані, у зв’язку з тим, що вони не покривають усі потреби діяльності навчального центру. Тому для оптимізації роботи НВЦА запропоновано реалізувати нову систему для поставлених задач, що з одного боку діє як самостійна система для складання розкладу, а з іншого – є модулем CRM підсистеми для НВЦА, що також підтримує інтеграцію з розкладом університету CIST. Дана взаємодія сервісів допоможе запобігти накладанням пар у розкладах навчального центру та університету, оптимізує використання ресурсів навчальних аудиторіях шляхом надання можливості ділити одну аудиторію для різних груп у разі неповної її заповненості студентами з «основної» групи.

1. Аллен Д. Как привести дела в порядок. Искусство продуктивности без стресса [Текст] / Д. Аллен — М.: Манн, Иванов и Фербер, 2011. — 368 с.
2. НВЦА [Електронний ресурс] / Сайт НВЦА. - Режим доступу: : www/ URL : https://ostpc.org.ua/ – Загол. з екрану.

3. Синило, Л. Сложнее, чем кажется — внедрение CRM [Текст] / Л. Синило. – М.: Новый Маркетинг, 2006. — 500с.

4. Prosperworks [Електронний ресурс] / Сайт сервісу Рrosperworks. - Режим доступу: : www/ URL : https://www.prosperworks.com/. – Загол. з екрану.

5. ZoroCRM – CRM that helps you sell [Електронний ресурс] / Сайт сервісу ZoroCRM. - Режим доступу: : www/ URL : https://www.zoho.eu/crm/. – Загол. з екрану.

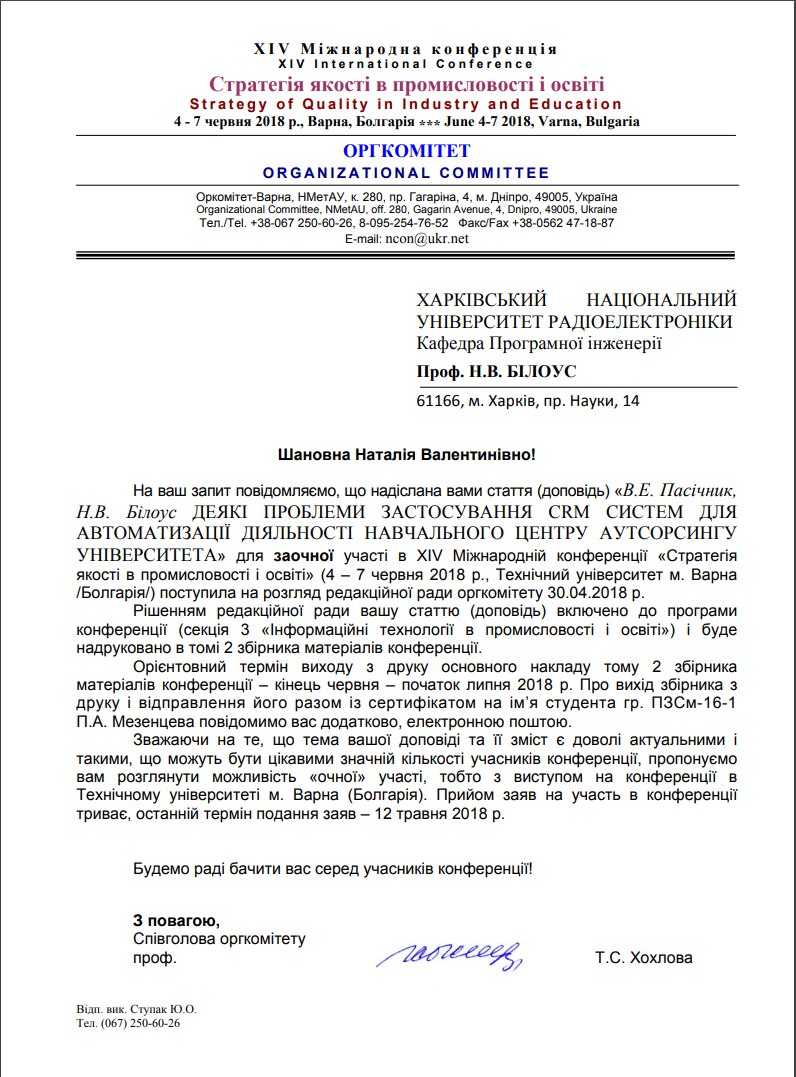
6. Бізнес працює в Bitrix24 [Електронний ресурс] / Сайт сервісу Bitrix24. - Режим доступу: : www/ URL : https://www.bitrix24.ua/. – Загол. з екрану.

7. Розклад ХНУРЕ СIST [Електронний ресурс] / Сайт сервісу СIST. - Режим доступу: : www/ URL : https:// cist.nure.ua/. – Загол. з екрану.

5

**ДОДАТОК Б**

Лист від оргкомітету XIV Міжнародної конференції «Стратегія якості в промисловості та освіті»



**ДОДАТОК В**

Приклади програмного коду

Нижче приведено код сервісу для отримання даних з CIST API:

**@Injectable()**

**export class CistService {**

**constructor(private http: HttpClient) { }**

**getCenterAuditoriesSchedule(): Observable<CistAuditorySchedule> {**

**if (localStorage.getItem('auditories-schedule')) {**

**return of(JSON.parse(localStorage.getItem('auditories-schedule')));**

**}**

**return forkJoin([**

**this.http**

**.get<CistAuditorySchedule>(`${CIST\_API\_URL}${AUDITORY\_URL}?p\_id\_auditory=${id455a}`),**

**this.http**

**.get<CistAuditorySchedule>(`${CIST\_API\_URL}${AUDITORY\_URL}?p\_id\_auditory=${id114i}`)**

**])**

**.pipe(**

**map((response: CistAuditorySchedule[]): CistAuditorySchedule => {**

**const response455a = response[0];**

**const response114i = response[1];**

**const formattedEvents455a: CalendarEvent[] = [];**

**const formattedEvents114i: CalendarEvent[] = [];**

**(response455a.events as CistAuditoryEvent[]).forEach((event: CistAuditoryEvent) => {**

**formattedEvents455a.push(this.formatCalendarEvent(event, '#ff7600'));**

**});**

**(response114i.events as CistAuditoryEvent[]).forEach((event: CistAuditoryEvent) => {**

**formattedEvents114i.push(this.formatCalendarEvent(event, '#ffcf00'));**

**});**

**localStorage.setItem('auditories-schedule', JSON.stringify({**

**...response,**

**events: [...formattedEvents455a, ...formattedEvents114i]**

**}));**

**return {**

**...response,**

**events: [...formattedEvents455a, ...formattedEvents114i]**

**};**

**})**

**);**

**}**

**getAuditorySchedule(id: number): Observable<CistAuditorySchedule> {**

**if (localStorage.getItem(`schedule-auditory-${id}`)) {**

**return of(JSON.parse(localStorage.getItem(`schedule-auditory-${id}`)));**

**}**

**return this.http**

**.get<CistAuditorySchedule>(`${CIST\_API\_URL}${AUDITORY\_URL}?p\_id\_auditory=${id}`)**

**.pipe(**

**tap((res) => localStorage.setItem(`schedule-auditory-${id}`, JSON.stringify(res)))**

**);**

**}**

**getGroups() {**

**if (localStorage.getItem('groups')) {**

**return of(JSON.parse(localStorage.getItem('groups')));**

**}**

**return this.http**

**.get(`${CIST\_API\_URL}${GROUPS\_URL}`)**

**.pipe(**

**map((response) => {**

**const allGroups = {};**

**const allFaculties = response['university'].faculties;**

**allFaculties.forEach((faculty) => {**

**let facultyGroups = [];**

**faculty.directions.forEach((direction) => {**

**if (direction.specialities && direction.specialities.length) {**

**direction.specialities.forEach((speciality) => {**

**if (speciality.groups) {**

**facultyGroups = [...facultyGroups, ...speciality.groups];**

**}**

**});**

**} else {**

**facultyGroups = [...facultyGroups, ...direction.groups];**

**}**

**allGroups[faculty.id] = facultyGroups;**

**});**

**});**

**localStorage.setItem('groups', JSON.stringify({allFaculties, allGroups}));**

**return {allFaculties, allGroups};**

**})**

**);**

**}**

**getTeacherSchedule(id: number) {**

**if (localStorage.getItem(`schedule-teacher-${id}`)) {**

**return of(JSON.parse(localStorage.getItem(`schedule-teacher-${id}`)));**

**}**

**return this.http**

**.get(`${CIST\_API\_URL}${SCHEDULE\_URL}?timetable\_id=${id}${teacherScheduleTypeParam}`)**

**.pipe(**

**tap((res) => localStorage.setItem(`schedule-teacher-${id}`, JSON.stringify(res)))**

**);**

**}**

**getGroupSchedule(id: number) {**

**if (localStorage.getItem(`schedule-group-${id}`)) {**

**return of(JSON.parse(localStorage.getItem(`schedule-group-${id}`)));**

**}**

**return this.http**

**.get(`${CIST\_API\_URL}${SCHEDULE\_URL}?timetable\_id=${id}`)**

**.pipe(**

**tap((res) => localStorage.setItem(`schedule-group-${id}`, JSON.stringify(res)))**

**);**

**}**

**private formatCalendarEvent(event: CistAuditoryEvent, color: string) {**

**return {**

**start: moment.unix(<number>event.start\_time).format(),**

**end: moment.unix(<number>event.end\_time).format(),**

**title: event.auditory,**

**editable: false,**

**color**

**};**

**}**

**}**

Нижче приведено код темплейту для форми створення запиту на додання події до розкладу:

**<div mat-dialog-content class="dialog-wrapper">**

**<h2>New event request</h2>**

**<p><span class="bold">Teacher: </span> Bilous N.V.**

**<p class="warning" \*ngIf="collisionsWithTeacher">**

**This event has conflict with your university schedule**

**</p>**

**<p class="warning" \*ngIf="collisionsWithTeacherCenter">**

**This event has conflict with your OSTPC schedule**

**</p>**

**<hr/>**

**<form>**

**<mat-form-field>**

**<input matInput placeholder="Event description" required**

**[formControl]="scheduleRequestForm?.get('description')">**

**</mat-form-field>**

**<hr/>**

**<p class="warning" \*ngIf="collisionsWithAuditory">**

**This auditory is already reserved by university**

**</p>**

**<p class="warning" \*ngIf="collisionsWithAuditoryCenter">**

**This auditory is already reserved by OSTPC**

**</p>**

**<mat-form-field>**

**<mat-select placeholder="Select auditory" [formControl]="scheduleRequestForm?.get('auditoryId')">**

**<mat-option \*ngFor="let auditory of auditoriesList" [value]="auditory.id">**

**{{ auditory.name }}**

**</mat-option>**

**</mat-select>**

**</mat-form-field>**

**<mat-radio-group [formControl]="scheduleRequestForm?.get('isSingleEvent')">**

**<mat-radio-button [value]="true">Single event</mat-radio-button>**

**<mat-radio-button [value]="false">Recurrent events</mat-radio-button>**

**</mat-radio-group>**

**<hr/>**

**<mat-form-field \*ngIf="!scheduleRequestForm?.get('isSingleEvent').value">**

**<input matInput placeholder="Number of events" type="number" min="1"**

**[formControl]="scheduleRequestForm?.get('numberOfEvents')">**

**</mat-form-field>**

**<div class="time-selecting-block">**

**<mat-form-field class="datepicker-block">**

**<input matInput [matDatepicker]="dp"**

**placeholder="Select start date"**

**[formControl]="scheduleRequestForm?.get('startDate')">**

**<mat-datepicker-toggle matSuffix [for]="dp"></mat-datepicker-toggle>**

**<mat-datepicker #dp disabled="false"></mat-datepicker>**

**</mat-form-field>**

**<div class="time-block">**

**<label for="startTime">Start time</label>**

**<input atp-time-picker readonly id="startTime"**

**[formControl]="scheduleRequestForm?.get('startTime')"/>**

**</div>**

**</div>**

**<p class="warning" \*ngIf="areStartEndTimeInvalid">Start of the event should be before its end</p>**

**<div class="time-selecting-block">**

**<mat-form-field class="datepicker-block">**

**<input matInput [matDatepicker]="dp1"**

**placeholder="Select end date"**

**[formControl]="scheduleRequestForm?.get('endDate')">**

**<mat-datepicker-toggle matSuffix [for]="dp1"></mat-datepicker-toggle>**

**<mat-datepicker #dp1 disabled="false"></mat-datepicker>**

**</mat-form-field>**

**<div class="time-block">**

**<label for="endTimeSingle">End time</label>**

**<input atp-time-picker readonly id="endTimeSingle"**

**[formControl]="scheduleRequestForm?.get('endTime')"/>**

**</div>**

**</div>**

**<hr/>**

**<mat-checkbox [formControl]="scheduleRequestForm?.get('checkUniversityGroupSchedule')">**

**Check on conflicts with university group schedule**

**</mat-checkbox>**

**<p class="warning" \*ngIf="groupIsNotFromTheList">Please select group from the list</p>**

**<p class="warning" \*ngIf="collisionsWithGroup"> This event has conflict with this group university schedule</p>**

**<div class="group-block">**

**<mat-form-field \*ngIf="scheduleRequestForm?.get('checkUniversityGroupSchedule').value">**

**<mat-select placeholder="Select faculty" [formControl]="scheduleRequestForm?.get('facultyGroupId')">**

**<mat-option \*ngFor="let faculty of universityFacultiesList" [value]="faculty.id">**

**{{ faculty.full\_name }}**

**</mat-option>**

**</mat-select>**

**</mat-form-field>**

**<mat-form-field \*ngIf="scheduleRequestForm?.get('checkUniversityGroupSchedule').value && selectedGroupsList.length">**

**<input type="text" placeholder="Select group" matInput**

**[formControl]="scheduleRequestForm?.get('universityGroupId')"**

**[matAutocomplete]="auto">**

**<mat-autocomplete #auto="matAutocomplete" [displayWith]="displayFn.bind(this)">**

**<mat-option \*ngFor="let option of filteredGroupOptions | async" [value]="option.id">**

**{{ option.name }}**

**</mat-option>**

**</mat-autocomplete>**

**</mat-form-field>**

**</div>**

**</form>**

**</div>**

**<div mat-dialog-actions class="popup-controls">**

**<button mat-raised-button (click)="onNoClick()">Cancel</button>**

**<button mat-raised-button color="primary" (click)="sendAddRequest()">**

**Send adding schedule request**

**</button>**

**</div>**

**ДОДАТОК Г**

Слайди презентації

