МІНІСТЕРСТВО ОСВІТИ І НАУКИ,

МОЛОДІ ТА СПОРТУ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

Кафедра технічної кібернетики

ПОЯСНЮВАЛЬНА ЗАПИСКА ДО КУРСОВОГО ПРОЕКТУ

з дисципліни «Управління проектами»

на тему:

«Розробка мобільного клієнту: актуальний розклад»

«ВИКОНАВ»

ст. групи ІК-21

Копилаш В.А.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Підпис Дата

«ПРИЙНЯВ» з оцінкою

Мелкумян К. Ю.

Керівник КП

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Підпис Дата

м.Київ

Національний технічний університет України “КПІ”

(назва вищого навчального закладу)

Кафедра Технічної кібернетики

Дисципліна Управління проектами

Спеціальність 6.050201 «Комп’ютеризовані та робототехнічні системи»

Курс **IV** Група **ІК-21** Семестр **I**

ЗАВДАННЯ

**на курсовий проект студента**

Копилаша Владислава Андрійовича

(прізвище, ім’я, по батькові)

1. Тема роботи Розробка мобільного клієнту: актуальний розклад\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

2. Строк здачі студентом закінченої роботи

3. Вихідні дані до роботи технічне завдання на розробку мобільного додатку для

відображення актуального розкладу

4. Зміст розрахунково – пояснювальної записки (перелік питань, які підлягають розробці)

Склад та ієрархія задач та підзадач, що вирішуються в додатку, схеми вхідних та вихідних

інформаційних потоків, специфікація процесів, інформаційне забезпечення додатку,

#### фільтри, запити додатку, інтерфейси користувача, інструментарій розробки

5. Перелік графічного матеріалу ( з точним зазначенням обов’язкових креслень)

схема БД, екранні форми

6. Режим доступу до ресурсу: http://edisk.ukr.net/?do=dir#cdir=i106853422

7. Дата видачі завдання 05.10.2015

**КАЛЕНДАРНИЙ ПЛАН**

|  |  |  |  |
| --- | --- | --- | --- |
| **№**  п\п | **Назва етапів курсового проекту** | **Строк виконання**  **етапів роботи** | **Примітки** |
| **1** | Визначення теми | 19.09.2015 |  |
| **2** | Обробка літератури по темі | 28.09.2015 |  |
| **3** | Створення структури програми | 01.10.2015 |  |
| **4** | Визначення попереднього алгоритму | 15.10.2015 |  |
| **5** | Розробка попереднього інтерфейсу | 30.10.2015 |  |
| **6** | Написання основного коду програми | 10.11.2015 |  |
| **7** | Оптимізація і тестування | 25.11.2015 |  |
| **8** | Наладка програми та фінальна компіляція | 01.12.2015 |  |
| **9** | Оформлення роботи | 15.12.2015 |  |
| **10** | Захист курсового проекту |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Студент Копилаш В.А.

(прізвище, ім’я, по батькові студента)

Керівник Мелкумян К.Ю.

( прізвище, ім’я, по батькові викладача)

« » 2015 р.

Зміст

[Вступ 5](#_Toc438417919)

[Склад та ієрархія задач 6](#_Toc438417920)

[Схеми вхідних та вихідних інформаційних потоків 7](#_Toc438417921)

[Специфікація процесів 9](#_Toc438417922)

[Інформаційне забезпечення додатку 11](#_Toc438417923)

[Фільтри, запити, звіти додатку 15](#_Toc438417924)

[Інструментарій розробки 19](#_Toc438417925)

[Інструкція користувачу 22](#_Toc438417926)

[Вимоги до апаратного забезпечення 32](#_Toc438417927)

[Висновок 33](#_Toc438417928)

[Список джерел 34](#_Toc438417929)

# ****Вступ****

З поширенням мобільних пристроїв доля ринку мобільних застосунків зростає кожного дня. Люди все менше використовують десктопні та веб-додатки, а надають перевагу сучасним смартфонам та планшетам. Це надає певну гнучкість студентові, однак він повинен завжди під рукою мати щось на кшталт органайзера. Головним заняттям студента є навчання, тож досить важливо мати швидкий доступ до розкладу занять. Однак, досить часто відбуваються різні зміни: викладач переніс пару, розклад не є актуальним – тому потрібно мати можливість скорегувати власний розклад. Саме в цьому полягає актуальність теми: завжди мати релевантний розклад занять, щоб правильно планувати свій час та швидко отримувати всю необхідну інформацію щодо предмету.

*Метою даної роботи* є проектування мобільного клієнту, за допомогою якого користувач зможе переглядати розклад занять та за необхідності корегувати.

# ****Склад та ієрархія задач****

Спроектувати додаток для Android, що дозволить переглядати актуальний розклад занять та корегувати його.

1. Вибрати підхід до розв’язку проблеми;
2. Спроектувати БД;
3. Вибрати архітектуру;
4. Спроектувати інтерфейси (Дизайн в стилі Google Material);
5. Реалізувати функціонал:
   1. Виведення розкладу;
   2. Зміна тижня;
   3. Вибір групи;
   4. Обробка помилок.

# ****Схеми вхідних та вихідних інформаційних потоків****

У додатку наявні декілька інформаційних процесів: отримання розкладу певної групи студентів, отримання розкладу викладача, авторизація викладача, зміна розкладу.



Рис. 1. Інформаційні потоки для процесу отримання розкладу групи

Вхідною величиною для процесу отримання розкладу групи є шифр групи, вихідною – розклад групи, обов’язковою умовою є наявність інтернет-з’єднання.

Майже аналогічним є інформаційний процес отримання розкладу викладача, тільки на вхід подається ПІБ викладача.



Рис. 2. Інформаційні потоки для процесу отримання розкладу викладача

Для процесу авторизації викладача вхідні та вихідні інформаційні потоки мають наступний вигляд. На вхід подаються логін та пароль викладача, після успішної авторизації сервер створює сесію для даного користувача та зберігає інформацію про викладача в ній.



Рис. 3. Інформаційні потоки для процесу авторизації викладача

Процес перенесення пари має наступні входи та виходи: інформація про зміни (id пари, дата, з якої переноситься, дата, на котру переноситься пара, прапорець, якщо пара відміняється взагалі, та номер пари на яку переноситься) ­подається на вхід, на виході отримуємо статус про зміну (успішно чи ні).



Рис. 4. Інформаційні потоки для процесу зміни розкладу

# ****Специфікація процесів****

1. Специфікація процесу авторизації користувача

**ЯКЩО** логін вірний **І** пароль вірний **І** доступне підключення до інтернету

**ТО** створити на сервері сесію **І** записати в неї id користувача

**ІНАКШЕ** вивести помилку

1. Специфікація процесу завантаження розкладу викладача

**ЯКЩО** доступне підключення до інтернету **І** знайдено викладача з введеним ПІБ

**ТО**

**ЯКЩО** є зміни в розкладі **ТО** завантажити розклад зі змінами

**ІНАКШЕ** завантажити розклад

**ІНАКШЕ** вивести повідомлення про помилку.

1. Специфікація завантаження розкладу

**ЯКЩО** шифр групи введено вірно **І** доступне підключення до інтернету

**ТО**

**ЯКЩО** є зміни в розкладі **ТО** завантажити розклад зі змінами

**ІНАКШЕ** завантажити розклад

**ІНАКШЕ** вивести повідомлення про помилку.

1. Специфікація процесу зміни розкладу

**ЯКЩО** користувач авторизований **І** має доступ до мережі інтернет

**ТО**

**ЯКЩО** пара, в день з вказаною датою, вільна

**ТО** перенести пару з вказаної дати на іншу дату, вказану користувачем

**ІНАКШЕ** вивести повідомлення про те, що пара вже зайнята

**ІНАКШЕ** вивести помилку

1. Специфікація зміни тижня

**ЯКЩО** вибрано перший тиждень

**ТО** вивести розклад для першого тижня

**ІНАКШЕ** вивести розклад для другого тижня.

1. Специфікація вибору дня тижня

На основі натиснутої вкладки, вивести розклад для певного дня тижня.

# ****Інформаційне забезпечення додатку****

Сучасні підходи до розробки архітектури дуже відрізняються. Дуже популярною є 3-рівнева архітектура клієнт-сервер. Вона складається з рівня БД, рівня бізнес логіки та рівня представлення.

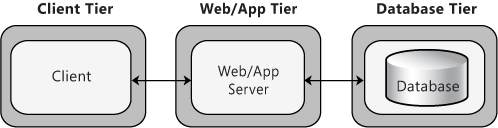


Рис. 5. Тьохрівнева архітектура

Розв’язок проблеми актуальності розкладу полягає в можливості його корегування. Для цього необхідно зберігати не тільки розклад пар, а ще й мати додаткову таблицю для записів про перенесення чи відміни пар. Тому було спроектовано базу даних, що містить всі необхідні таблиці та зв’язки, її схема приведена в додатку А. БД складається з 6 таблиць.

Таблиця teachers призначена для збереження даних про викладачів та має наступні поля:

* id – primary key
* name – ім’я викладача
* surname – прізвище викладача
* middleName – ім’я викладача
* login – логін
* password – хеш пароля

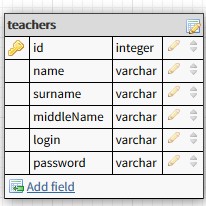


Рис. 6. Таблиця teachers

Таблиця subjects призначена для збереження даних про предмети, вона має наступні поля:

* id – primary key
* name – назва предмету

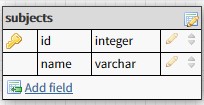


Рис. 7. Таблиця subjects

Таблиця groups призначена для збереження даних про групи, дана таблиця має наступні поля:

* id – primary key
* code – шифр групи

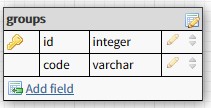


Рис. 8. Таблиця groups

Найважливішою таблицею є таблиця lessons, оскільки в ній фактично зберігається розклад занять, вона пов’язує між собою викладачів і предмети.

Атрибути цієї таблиці:

* id – primary key
* type – тип заняття (лекція, практика та ін.)
* day – день тижня
* lessonNumber – номер пари
* subjectId – foreign key references subjects(id)
* teacherId – foreign key references teachers(id)
* room – номер аудиторії

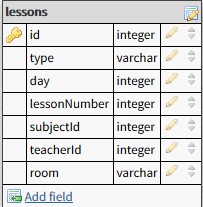


Рис. 9. Таблиця lessons

Один викладач може читати багато предметів, тому маємо зв’язок 1 до багатьох, аналогічно з таблицею subjects, бо один предмет може мати багато занять.

Так як декілька груп можуть мати спільне заняття, тобто між заняттями та групами зв’язок багато до багатьох, треба ввести додаткову розв’язочну таблицю groups\_lessons, вона має такі поля:

* lessonId – foreign key references lessons(id)
* groupId – foreign key references groups(id)

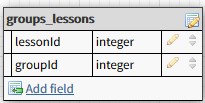


Рис. 10. Таблиця groups\_lessons

За допомогою записів у даній таблиці можна унікально ідентифікувати пару група-заняття.

Остання таблиця lesson\_changes призначення для збереження записів про редагування розкладу, вона має наступні атрибути:

* id – primary key
* dateFrom – дата, з якої переносять заняття
* dateTo – дата, на яку переносять заняття
* lessonNumber – номер пари, на яку переносять заняття
* lessonId – foreign key references lessons(id)
* isRemoved – прапорець, що вказує на те, що пара взагалі відміняється

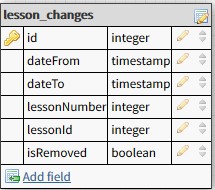


Рис. 11. Таблиця lessons\_changes

Кожен запис в даній таблиці показує зміну однієї пари з поточного розкладу. Таким чином, вибірка розкладу з БД завжди буде враховувати зміни, внесені викладачем.

# Фільтри, запити, звіти додатку

Щоб отримати дані про розклад занять треба робити запити до БД. Дуже доречно було б підключити ORM та сконфігурувати моделі, що повністю б відображали відношення між сутностями системи за допомогою класів. Якщо ж використовувати прості SQL-запити, то вони б виглядали наступним чином:

1. Запит на пошук викладача за логіном та хешем пароля

select \* from teachers where login = <логін> AND password = <хеш\_пароля>

1. Запит на вивантаження розкладу викладача

select \* from lessons

join teachers on lessons.teacherId = teachers.id

join subjects on lessons.subjectId = subjects.id

join groups\_lessons on groups\_lessons.lessonId = lessons.id

join groups on groups\_lessons.groupId = groups.id

where lessons.name = <ім’я викладача> AND teachers.surname = <прізвище викладача> AND teachers.middleName = <по-батькові викладача>

group by lesson.id

order by lessons.day;

1. Запит на вивантаження розкладу групи

select \* from lessons

join teachers on lessons.teacherId = teachers.id

join subjects on lessons.subjectId = subjects.id

join groups\_lessons on groups\_lessons.lessonId = lessons.id

join groups on groups\_lessons.groupId = groups.id

where groups.code = <шифр\_групи>

order by lessons.day;

1. Запит на вибірку змін розкладу

Так як розклад відображається на період 2 тижнів, то запит повинен вибирати дані за цей період.

select \* from lesson\_changes

join lessons on lesson\_changes.lessonId = lessons.id

join subjects on lessons.subjectId = subjects.id

join teachers on lessons.teacherId = teachers.id

where (dateFrom between <дата\_початку\_періоду> AND <дата\_кінця\_періоду>) OR (dateTo between <дата\_початку\_періоду> AND <дата\_кінця\_періоду>);

1. Запит на запис змін до розкладу

insert into lesson\_changes(dateFrom, dateTo, lessonNumber, lessonId, isRemoved) values (<>);

1. Запит на перевірку вільності пари в поточному розкладі

select \* from lessons

join groups\_lessons on lessons.id = group\_lessons.lessonId

join groups on groups\_lessons.groupId = groups.id

where groups.code = <шифр\_групи> AND

lessons.day = <день, на який переноситься заняття> AND lessons.lessonNumber = <номер пари, на яку переноситься заняття>

1. Запит на перевірку вільності пари, враховуючи зміни

select \* from lesson\_changes where dateTo = <дата, на яку переноситься заняття> AND lessonNumber = <номер пари, на яку переноситься заняття>

Сервер додатку має надавати REST API для отримання інформації про розклад занять, його зміну і т.д.

**REST** (скор. англ. *Representational State Transfer*) — підхід до архітектури мережевих протоколів, які забезпечують доступ до інформаційних ресурсів. Був описаний і популяризований у 2000 році Роєм Філдінгом (Roy Fielding), одним із творців протоколу HTTP. Найвідомішою системою, побудованою переважно за архітектурою REST, є сучасна Всесвітня павутина. Дані повинні передаватися у вигляді невеликої кількості стандартних форматів (наприклад HTML, XML, JSON). Мережевий протокол (як і HTTP) повинен підтримувати кешування, не повинен залежати від мережевого прошарку, не повинен зберігати інформацію про стан між парами «запит-відповідь». Стверджується, що такий підхід забезпечує масштабування системи і дозволяє їй еволюціонувати з новими вимогами.

Враховуючи вимоги до проектування REST-сервісів було спроектовано сервіси, що підтримують наступні запити (обмін здійснюється за допомогою формату json):

1. Запит для отримання розкладу групи

GET <http://localhost/groups/:code/timetable>

1. Запит для отримання розкладу викладача

GET <http://localhost/teachers/:name/timetable>

1. Запит на авторизацію

POST <http://localhost/auth>

В body передається 2 параметри: login, password

1. Запит на внесення змін до розкладу

PUT <http://localhost/teachers/:id/lessons/id>

В body передаються дані про зміну заняття, наприклад:

{

“dateFrom”: “Wed Dec 16 2015 17:49:49 GMT+0200”;

“dateTo”: "Wed Dec 18 2015 17:49:49 GMT+0200”;

“lessonNumber”: 3;

“isRemoved”: false;

}

Клієнт звертатиметься за допомогою HTTP запитів до API, витягуватиме потрібні дані та виводитиме їх на екран.

# Інструментарій розробки

Так як основною платформою було вибрано Android, розробка під який апріорі можлива лише на мові програмування Java та є дуже специфічною, були знайдені альтернативні шляхи розробки мобільних додатків.

Розробка мобільних додатків на сучасному етапі є вкрай популярною. Кількість користувачів мобільних пристроїв на різних платформах зростає з кожним днем. Сьогодні користувач вимагає широкого спектру різноманітного програмного забезпечення для офісу та вирішення бізнес-задач. Також, досить велика частка програмного ринку належить ігровим додаткам для пристроїв на базі iOS і Android платформ.

**Існує три види мобільних додатків:**

* **Веб-додатки, сайти**. Найпопулярніший тип додатків. Сучасні мобільні пристрої надають користувачеві, який працює в мережі Інтернет, такі ж можливості, що й стаціонарні комп’ютери. Обумовлено це підтримкою HTML 5. Окрім того, додатки для мобільних пристроїв це ідеальний варіант для швидкого старту з невеликими вкладеннями і відмінною функціональністю. Мобільні сайти – це рішення для всіх без винятку платформ.
* **Гібридне ПЗ.** Це можливість доступу до всього функціоналу мобільних пристроїв, при цьому для їх створення використовуються фреймворки, HTML, Java, CSS. Це нативне ПЗ з інтегрованим мовою розмітки. Такі додатки – чудове рішення для тих, хто бажає використовувати всі переваги нативних додатків в поєднанні з останніми розробками в веб-технологіях. Прикладом гібридного програмного забезпечення може служити Facebook, що завантажується з магазину ПЗ, але потребує підключення до Інтернету.
* **Нативне ПЗ**. Найвимогливіший до ресурсів вид додатків. У той же самий час, дозволяє скористатися всіма можливостями операційної системи.
* Це найбільш функціональний і продуктивний вид мобільних додатків.

На частку мобільних пристроїв на базі Android припадає 50 відсотків всіх користувачів. Створення додатків для цієї платформи відрізняється від розробки сайту. В даному випадку також приділяється увага інтерфейсу – дружньому та інтуїтивно зрозумілому. Щоб не зникало бажання використовувати додаток кожен день, важливо опрацювати прототипи і дизайн програми. Розробка додатків для Аndroid вимагає від розробників глибоких знань і розуміння принципів ефективної взаємодії користувача з додатком, законів дизайну.

Суттю гібридного застосування є те, що це веб-контент, а в смартфоні відкривається оболонка, що відображає цей контент. Найпопулярнішим фреймворком для гібридних додатків є Apache Cordova, що слугує мостом між функціями системи та веб-браузером, що в термінології гібридних застосувань називається Webview. Цей фреймворк є безкоштовним та розповсюджується за допомогою npm (node package manager). Цей спосіб підходить для людей, що дуже люблять все налаштовувати власноруч. Однак, на основі Cordova існують такі SaaS як PhoneGap (створений компанією Adobe) та Cocoon.io, який після деяких досліджень було вибрано як основний інструмент для розробки мобільного додатку. Серед переваг даного фреймворку висока швидкодія за рахунок технології Webview+ та відсутності проблем з кротплатформеністю, що забезпечується Chromium з його віртуальною машиною V8. Тобто орієнтуючись на Webkit, ми завжди будемо мати однакове відображення на всіх пристроях.

Також було вирішено писати клієнтську частину на React.js, що розробляється корпорацією Facebook, та є дуже швидким. У якості бібліотеки для HTTP запитів було використано whatwg-fetch, що розвивається командою Github.

Було поставлено завдання про Google Material UI. Було застосовано фреймворк material-ui, який є дуже популярним та відповідає всім гайдлайнам, щодо дизайну. Таким чином ми заклали user friendly інтерфейс ще на початку проекту, при виборі технології та проектуванні.

У якості VCS було вибрано Github. Посилання на репозиторій: <https://github.com/kopylash/timetable/tree/dev>

# Інструкція користувачу

В даному розділі буде детально описано роботу додатку з приведенням всіх екранних форм. При відкритті додатку користувач бачить екран, зображений на рисунку 12.

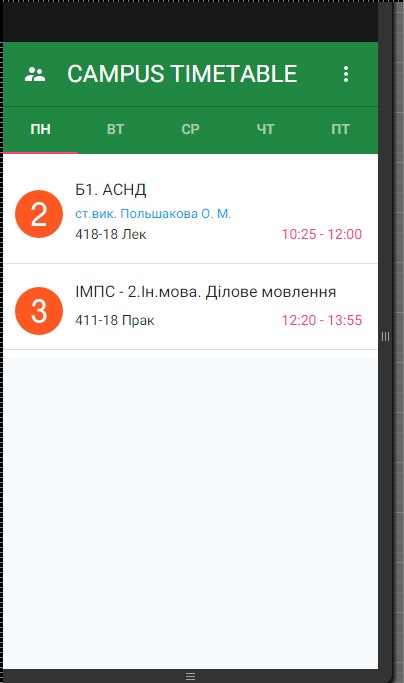


Рис. 12 Початковий екран

Можна побачити, що екран має панель вкладок зі списком днів, що генерується автоматично на основі даних, що прийшли від сервера. Є іконки меню та зміни групи. Під кожною вкладкою є список пар на цей день (рис. 13). Користувач може побачити назву предмету, прізвище та ініціали викладача, аудиторію та тип заняття і час пари. Зліва від інформації розташована іконка в стилі material, що відображає номер пари та привертає увагу – це дає змогу швидко орієнтуватися.

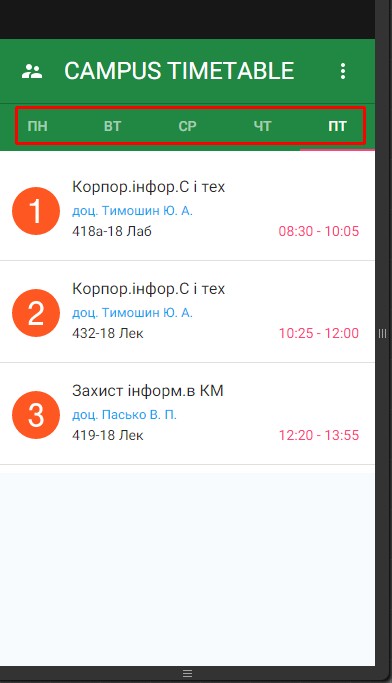


Рис. 13 Панель вкладок

Користувач може змінити тиждень натиснувши на кнопку меню у правому верхньому куті екрану. При чому одразу підсвічується поточний тиждень (рис. 14).

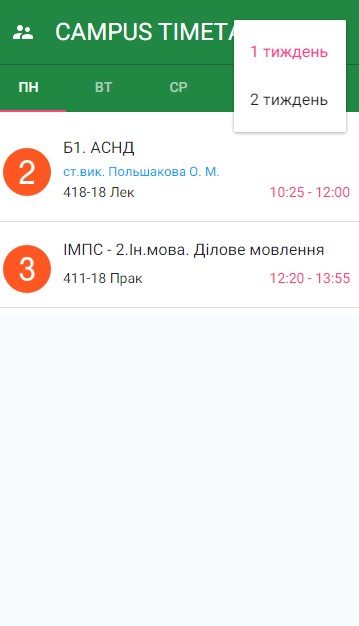


Рис. 14 Вибір тижня

Якщо вибрати інший тиждень, то порівнюючи з першим екраном, можна побачити, що розклад дійсно змінився і активним пунктом меню тепер є

«2 тиждень» (рис. 15)

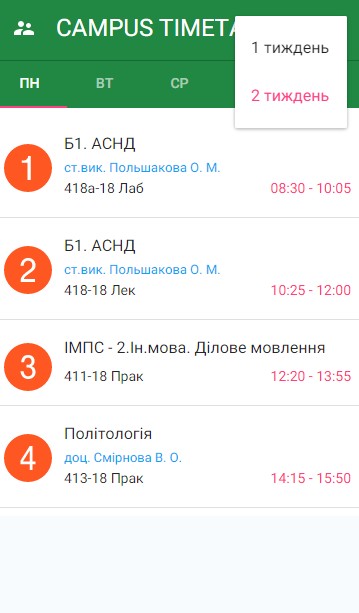


Рис. 15 Результат зміни тижня

Користувач може вибрати групу, для цього він натискає на іконку в лівому верхньому куті екрану і перед ним з’являється модальне вікно, де треба ввести шифр групи. Його можна вводити як кирилицею, так і латиницею з урахуванням всіх правил транслітерації (рис. 16).

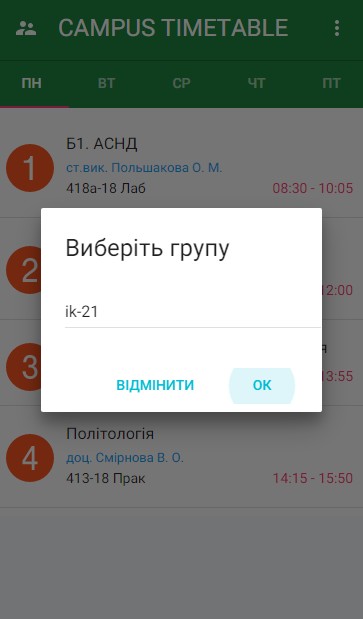


Рис. 16 Модальне вікно вибору групи

Для прикладу групу студентів змінюють на БЕ-21 (рис. 17).

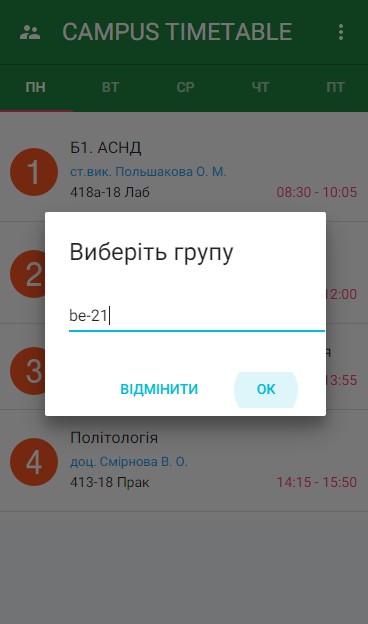


Рис. 17 Зміна групи на БЕ-21

Як результат можна побачити, що розклад змінився (рис. 18).

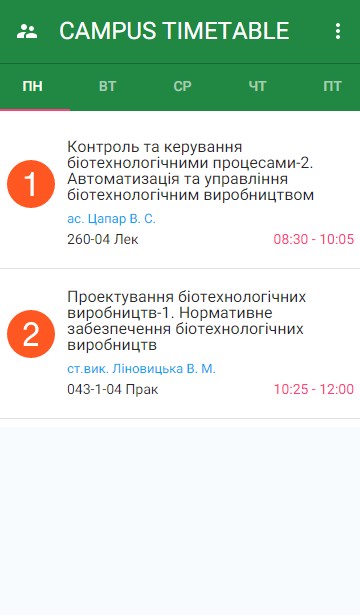


Рис. 18 Результат зміни групи на БЕ-21 – її розклад

Якщо знову відкрити модальне вікно вибору групи, то видно, що поточне значення групи встановлене як «be-21», що підтверджує успішність виконання попередніх команд (рис. 19).

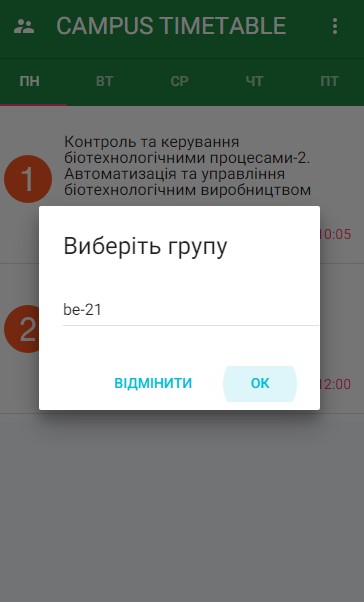


Рис. 19 Поточне значення групи

Якщо групу введено невірно, наприклад «ik-21ваів» (рис. 20), або вона не існує, або сервіс не працює, що малоімовірно, користувачу показується повідомлення про помилку (рис. 21).

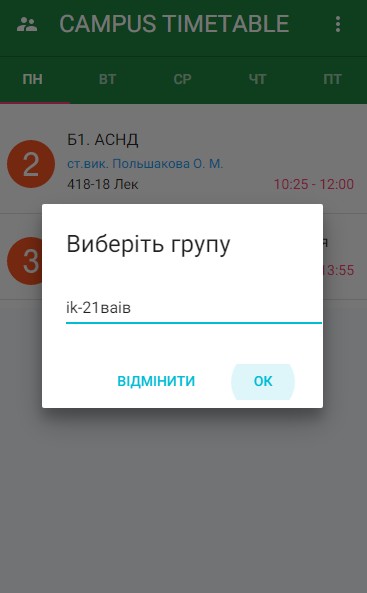


Рис. 20 Невірне введення шифру групи

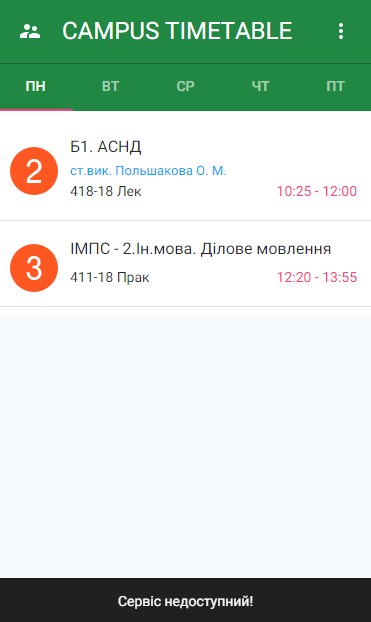


Рис. 21 Повідомлення про помилку

# 

# ****Вимоги до апаратного забезпечення****

***Системні вимоги:***

* *Операційна система:* Android 4.0 і вище;
* *ОЗП:* 512 Mb (рекомендовано 1024 Mb);
* *Процесор:* 800 Mhz (рекомендовано 1,2 Ghz);
* *Роздільна здатність екрану:* 1280х800;
* *Вільний простір на вінчестері :* 22 Мбайт;
* *Інтернет підключення;*
* Файл інсталяції – ***Campus Timetable.apk***.

# **Висновок**

Задачею даного проекту було створення мобільного клієнту для Android, що відображав би актуальний розклад. Провівши деякі дослідження, спроектувавши архітектуру на основі існуючих обмежень та внаслідок невеликої кількості часу на вивчення інших технологій був вибраний певний знайомий стек для розробки даного застосунку. В пошуках розв’язку проблеми було вирішено, що треба спроектувати базу даних, яка б дозволила зберігати інформацію про зміни у розкладі. Тоді досить легко відображати тимчасові зміни у розкладі. Також на виході маємо спроектовані REST-сервіси, які треба лише імплементувати. Вони охоплюють всі можливі варіанти запитів до сервера для отримання необхідної інформації чи збереження деяких даних. Було спроектовано інтерфейс користувача згідно з усіма рекомендаціями щодо UX, створено UI. Однією важливою особливістю є те, що даний проект легко розвивати та підтримувати. Facebook планує випустити React Native for Android – фреймворк, що дозволить уникнути посередника між вбудованими функціями пристрою та веб-інтерфейсом, він автоматично створює оригінальний байткод, що запускається JVM Android. Такий додаток працюватиме швидше, займатиме менше місця і його легко розвивати використовуючи Javascript. Також однією з перспектив є компіляція даного проекту під iOS, оскільки React Native підтримує кроссплатформеність, при чому елементи виглядають оригінально для кожної операційної системи, тобто не можна побачити timepicker Android в додатку для iOS. Поставлені задачі проекту було успішно виконано, на виході маємо мобільний клієнт для відображення актуального розкладу занять.

# Список джерел

1. Стаття з порталу Habrahabr. – Режим доступу: <http://habrahabr.ru/post/248799/> – Дата звернення: 20.09.2015;
2. Стаття з порталу Habrahabr. – Режим доступу: <http://habrahabr.ru/hub/ReactJS/> – Дата звернення: 20.09.2015;
3. Стаття з порталу Habrahabr. – Режим доступу: <http://habrahabr.ru/post/229629/> – Дата звернення: 20.09.2015;
4. Відео з конференції Facebook. – Режим доступу: <https://www.youtube.com/watch?v=KYzlpRvWZ6c> – Дата звернення: 30.09.2015;
5. Стаття з порталу Habrahabr. – Режим доступу: <http://habrahabr.ru/post/249107/> – Дата звернення: 15.10.2015;
6. Огляд React.js з прикладами. – Режим доступу: <http://tftf.ru/stati/javascript/reactjs/> – Дата звернення: 20.10.2015;
7. Бібліотека компонентів React.js. – Режим доступу: <http://react-components.com/>– Дата звернення: 01.11.2015;
8. Домашня сторінка Material UI. – Режим доступу: <http://www.material-ui.com/#/home> – Дата звернення: 01.11.2015;
9. Репозиторій Github проекту fetch. – Режим доступу: <https://github.com/github/fetch> – Дата звернення: 08.11.2015;
10. Сайт SaaS Cocoon.io – Режим доступу: <https://cocoon.io/> – Дата звернення: 08.11.2015;
11. Стаття з порталу Habrahabr. – Режим доступу: <http://habrahabr.ru/post/248799/> – Дата звернення: 09.11.2015;
12. Стаття з порталу Habrahabr. – Режим доступу: <https://cordova.apache.org/> – Дата звернення: 10.11.2015;
13. Документація плагіну Crosswalk. – Режим доступу: <https://crosswalk-project.org/documentation/cordova/cordova_4.html> – Дата звернення: 10.11.2015.