2. Що таке дизайн мислення: етапи, принципи, застосування, приклади / редакція Академії ITSTEP. *ITSTEP Академія. Online освіта.* 29.03.2023. URL: https://cloud.itstep.org/blog/what-is-design-thinking-and-how-is-it-used-in-modern-design (дата звернення: 15.05.2025).

3. Yarovyi D. Дизайн-мислення, Round Robin та брейншторми – як отримувати менше правок й ефективно розв'язувати завдання. *dou.ua.* 05.01.2023. URL: https://dou.ua/forums/topic/41463/ (дата звернення: 15.05.2025).

4. Сегментація цільової аудиторії за методом 5W Марка Шеррінгтона. *EdgeLab.* 24.11.2023. URL: https://edgelab.com.ua/blog/metod-5w-sherringtona-yak-i-chomu-treba-segmentuvaty-audytoriyu-medychnogo-brendu/ (дата звернення: 15.05.2025).

5. Gonul E. Styling Custom Components in Svelte. *Medium.* 14.12.2021. URL: https://erdemgonul.medium.com/styling-custom-components-in-svelte-3723ad752cd2 (date of access: 15.05.2025).

УДК 004.514

Куцмай В. Я., *здобувач вищої освіти 4 курсу спеціальності 122 Комп'ютерні науки,*
Антонов Ю. С., *канд. фіз.-мат. наук, доцент, доцент кафедри інформаційних технологій*

МОБІЛЬНИЙ ДОДАТОК ДЛЯ СИНХРОНІЗАЦІЇ ОСОБИСТОГО КАЛЕНДАРЯ ТА РОЗКЛАДУ ЗАНЯТЬ

Донецький національний університет імені Василя Стуса, м. Вінниця

Сучасні студенти часто витрачають час на ручне перенесення розкладу занять зі шкільного чи університетського порталу в особисті календарі. Такий процес є трудомістким і може допускати помилки. Водночас існують мобільні додатки або телеграм-боти для перегляду розкладу [1–2], але вони зазвичай лише відображають розклад без автоматичної синхронізації з особистим календарем. Переваги автоматичної синхронізації очевидні – зменшення ручної праці та своєчасне оновлення подій без помилок. Зазначимо, що в аналогічному випадку застосунок «Розклад занять в ПУЕТ» реалізував інтеграцію з Google Calendar API, що дозволяє автоматично додавати та видаляти події розкладу користувача.

Мета роботи – розробити Android-додаток, який забезпечить автоматичну синхронізацію особистого календаря користувача з розкладом занять. Для досягнення цієї мети реалізовано веб-API-з'єднання (Google Calendar API) для обміну даними між сервером розкладу та мобільним додатком. У структуру програми закладено принципи Clean Architecture, що розділяє логіку, відображення та дані на окремі шари, а також застосовано архітектурний патерн MVI (Model-View-Intent) для ефективної організації реактивного взаємодії користувацького інтерфейсу зі станом програми.

Методи дослідження та розробки включали аналіз наявних рішень, проєктування та прототипування. Проведено огляд літератури та програмних засобів у сфері мобільних календарів і синхронізації. На основі Clean Architecture організовано шари Domain, Data та Presentation, забезпечено залежності через Dependency Injection. Для графічного інтерфейсу застосовано декларативний фреймворк (Jetpack Compose) у поєднанні з MVI-патерном: це було виправдано потребою реактивного оновлення розкладу відповідно до даних сервера. Взаємодію з

Google Calendar реалізовано через REST-запити до Google Calendar API з авторизацією по OAuth, що гарантує безпечну передачу даних про події.

За результатами дослідження було створено прототип мобільного додатку для Android, який реалізує повний цикл автоматичної синхронізації навчального розкладу з особистим календарем користувача. Розроблений застосунок дає змогу студенту після автентифікації отримувати актуальний розклад занять з вебджерела, зберігати його локально в базі даних пристрою, а також експортувати події у форматі календаря Google. Архітектура додатка побудована на основі принципів Clean Architecture [3], що передбачає чітке розділення логіки на окремі шари (Presentation, Domain, Data). До того ж організації взаємодії з користувачем застосовано патерн MVI (Model-View-Intent), який забезпечує реактивне оновлення інтерфейсу у відповідь на зміну даних та подій.

Однією з ключових функціональних можливостей додатка є автоматична синхронізація з Google Calendar. Застосунок реалізує повноцінну взаємодію з Google Calendar API, що дає змогу додавати, змінювати або видаляти події розкладу без необхідності ручного введення. Події створюються на основі даних розкладу, зокрема із зазначенням дати, часу, назви дисципліни та місця проведення заняття. Передача інформації відбувається через захищене з'єднання з використанням OAuth 2.0. Такий підхід дає змогу користувачу бачити навчальні події безпосередньо у своєму календарі поряд з іншими особистими справами, що підвищує ефективність планування.

Крім технічної реалізації, було проведено тестування розробленого застосунку з метою оцінки його зручності, швидкодії й точності синхронізації. Порівняння з традиційним ручним способом перенесення розкладу показало істотну перевагу автоматизованого підходу: середній час, необхідний для додавання події в календар, зменшився майже вдвічі. За відгуками користувачів, також суттєво знизилася ймовірність помилок у введенні даних, а ризик пропуску занять через неуважність або застарілу інформацію в календарі був практично усунутий. Отже, результати підтвердили практичну доцільність запропонованого рішення та його потенціал для повсякденного використання студентами.

Отже, у роботі виконано аналіз проблеми ручного перенесення розкладу занять, обґрунтовано переваги Web-API-синхронізації та реалізовано прототип відповідного мобільного додатка. Використання Clean Architecture забезпечило зручність підтримки та розширення проєкту, а застосування MVI-патерну значно спростило управління станом інтерфейсу. Інтеграція з Google Calendar API гарантувала, що дані розкладу автоматично оновлюються у персональному календарі користувача. Практична перевірка показала, що новий додаток підвищує ефективність планування навчального процесу та зменшує рутинні операції, підтверджуючи доцільність обраних рішень.

Список використаних джерел

1. Козиряцький А. П. Мобільний додаток SKED. *Зв'язок.* 2018. № 1. С. 46–48.
2. Коляструк Б. А., Антонов Ю. С. Особливості розробки telegram бота для інформаційної підтримки навчального процесу. *Матеріали конференцій МЦНД.* 2020. С. 67–69. DOI: 10.36074/02.10.2020.v1.15.

3. Розробка мобільного android-додатку з застосуванням принципів clean architecture / Г. Козуб, Ю. Козуб, Г. Могильний, А. Жуков. *Вісник Східноукраїнського нац. ун-ту ім. Володимира Даля.* 2021. № 5(269). С. 5–10. URL: https://journals.snu.edu.ua/index.php/VisnikSNU/article/view/38

4. Козуб Ю., Козуб Г. Особливості розробки мультиплатформних застосунків на kotlin. *Вісник Хмельн. нац. ун-ту.* 2023. № 1. С. 224–229. URL: https://journals.khnu.km.ua/vestnik/?p=16772

УДК 004.056.523:004.651.54:004.439:681.5

Мисько Б. В., *здобувач вищої освіти 4 курсу спеціальності 122 Комп'ютерні науки,*
Фриз І. В., *канд. фіз.-мат. наук, старший викладач кафедри інформаційних технологій*

ІНТЕГРАЦІЯ JWT ТА АЛГОРИТМІВ ГЕШУВАННЯ ДЛЯ АВТЕНТИФІКАЦІЇ ТА АВТОРИЗАЦІЇ В CRM-СИСТЕМАХ

Донецький національний університет імені Василя Стуса, м. Вінниця

У сучасних інформаційних системах, зокрема у CRM-системах (Customer Relationship Management), безпека користувацьких даних та контроль доступу до ресурсів є критично важливими аспектами. З огляду на активне впровадження вебтехнологій та хмарних обчислень традиційні методи автентифікації, які базуються на збереженні паролів у відкритому вигляді або в сесійних змінних, втратили актуальність. Їм на зміну прийшли криптографічно стійкі алгоритми гешування та маркерні механізми авторизації, серед яких особливої уваги заслуговує JSON Web Token (JWT) [1].

Метою дослідження є розробка надійного механізму автентифікації та авторизації користувачів для CRM-системи. У межах роботи передбачено впровадження безпечного способу зберігання облікових даних, реалізацію цифрового токена для підтвердження ідентичності користувача, а також побудову гнучкої рольової моделі контролю доступу.

Гешування паролів – це незворотний процес, у якому зі вхідного значення (пароля) створюється фіксованої довжини геш-код. Одним із найбільш надійних алгоритмів є BCrypt, який має вбудований механізм сольової генерації та адаптивної складності [2]. Перевагами BCrypt є стійкість до атак методом перебору та можливість збільшення складності з часом.

JWT – відкритий стандарт (RFC 7519), який описує компактний, автономний спосіб безпечної передачі інформації між сторонами як JSON-об'єкт [1]. Токен складається з трьох частин: заголовка (header), корисного навантаження (payload) та підпису (signature). Його переваги включають можливість безсерверної авторизації, розширюваність і незалежність від конкретного протоколу передачі [3]. Архітектурна модель реалізації механізму автентифікації та авторизації базується на поєднанні трьох основних компонентів: