Всеукраїнській конкурс студентських наукових робіт з природничих, технічних та гуманітарних наук

Спеціальність: Інженерія програмного забезпечення

**Конкурсна робота**

**Шифр роботи «Proizd»**

**Мобільний додаток для замовлення проїзних карт студентами ВНЗ**

2019-2020 навчальний рік

**Зміст**

[Анотація 3](#_Toc33702935)

[Вступ 5](#_Toc33702936)

[1. Огляд та аналіз особливостей замовлення проїзних у ВНЗ 7](#_Toc33702937)

[1.1 Аналіз типів проїзних карт, доступних для замовлення в ВНЗ 7](#_Toc33702938)

[1.2 Основні положення та правила університету для щодо замовлення пільгових проїзних документів. 9](#_Toc33702939)

[1.3 Огляд існуючих варіантів замовлення проїзних квитків ВНЗ 11](#_Toc33702940)

[2. Розробка та опис функціонування додатку Proizd 13](#_Toc33702941)

[2.1 Розробка та специфікація вимог до програмного додатку Proizd 13](#_Toc33702942)

[2.2 Опис засобів програмної реалізації 16](#_Toc33702943)

[2.3 Архітектура додатку 19](#_Toc33702944)

[2.4 Опис функціонування додатку 24](#_Toc33702945)

[Висновки 28](#_Toc33702946)

[Література 29](#_Toc33702947)

# Анотація

Актуальність роботи визначається постійно зростаючим обсягом використання мобільних пристроїв. Сучасні додатки, що вирішують користувальницькі задачі, серед яких замовлення проїзних, та їх

оплата, обов’язково повинні мати представлення, яке дозволяє зручне використання програмою в мобільній версії.

**Мета** наукової роботи – підвищення зручності замовлення проїзних документів за допомогою мобільних пристроїв.

**Об’єкт** досліджень – інформатизація процесу замовлення проїзних.

**Предмет** досліджень – розробка програмних додатків для більш зручної взаємодії відповідальної за проїзні людини та усього ВНЗ.

**Завдання дослідження, які дозволяють досягти поставленої мети:**

1. Аналіз особливостей та структури елементів процесу оформлення проїзних ВНЗ.
2. Огляд існуючих мобільних додатків для замовлення проїзних.
3. Розробка та специфікація вимог до програмного додатку.
4. Аналіз засобів програмної реалізації.
5. Розробка архітектури системи та програмних модулів.

При виконанні досліджень та прийнятті рішень використовувались такі **підходи**: системний аналіз предметної галузі, структурування даних із використанням ментальних карт, моделювання програмного забезпечення за допомогою діаграми прецедентів UML, принципи об’єктно-орієнтованого програмування, архітектурний патерн Flux.

**Стислий опис результатів дослідження:** **реалізовано** програмний додаток для мобільних пристроїв під управлінням Android, який дозволяє швидко та зручно замовити проїзні студентам Державного університету телекомунікацій.

Робота має обсяг 29 аркушів, має 11 малюнків, 7 джерел посилань.

Ключові слова: проїзні, мобільний додаток, Flux, Android, React Native, Google Sheets, C#, ASP.Net Core.

# Вступ

Однією з важливих складових роботи ВНЗ є своєчасне оформлення проїзних для студентів. Значна частка студентів нашого вузу проживає в інших районах м.Києва та громадський транспорт, особливо метро, є невід’ємною та щоденною частиною маршруту до ВНЗ. Оскільки разовий проїзд у громадському транспорті є досить високим, існують студентські типи проїзних карт, ціна однієї поїздки в яких від 40% до 50% дешевша, в порівнянні зі звичайними.

Згідно з правилами КП, придбати пільговий проїзний на метро можливо лише через профспілки університетів. Що за собою тягне громіздкий та не завжди зручний процес.

Зараз, відповідальна за збір проїзних людина спершу, використовуючи соц. мережі чи телефонні дзвінки, формує списки людей, які хочуть отримати проїзний, після цього триває процес збору коштів за проїзні, який відбувається або за допомогою банківських переказів, або очної зустрічі із відповідальним та передачі готівки. І лише в кінці місяця за допомогою загальних зборів замовники отримують свої проїзні карточки.

На нашу думку, цей процес має значну кількість недоліків:

* Непотрібні витрати часу зі сторони відповідальної за проїзні особи
* Постійний контакт з замовниками, яких досить не мала кількість
* Велика кількість зустрічей та зборів

Сучасний підхід до замовлення проїзних документів вимагає його автоматизації. Це дозволить не тільки витрачати менше часу на весь процес, а й зробить його більш відкритим та зручним.

Ми пропонуємо автоматизувати збір заявок на проїзні, та їх підтвердження. Що дозволить відповідальній особі лише перевіряти список перед замовленням проїзних на ВНЗ та організовувати їх видачу.

В даній роботі реалізовано проект, який представляє собою безкоштовний мобільний додаток «Proizd», розміщений в Google Play Маркет, що дозволяє вільно використовувати його студентам та викладачам, що мають мобільні пристрої на базі операційної системи Android. Додаток «Proizd» реалізовано сучасною мовою Java Script із використанням фреймворка React Native та бібліотек AsyncStorage, XMLHttpRequest, UUID, Native Vector Icons.

Базові функції додатку дозволяють студенту в зручній формі сформувати своє замовлення проїзних на наступний місяць.

# Огляд та аналіз особливостей замовлення проїзних у ВНЗ

Процес замовлення проїзних є важливою складовою взаємодії студентів та адміністрації вищого навчального закладу. Не зважаючи на те, що проїзні всюди існують однакових типів, у нашому вузі є нюанси при їх замовленні.

## 1.1 [Аналіз типів проїзних карт, доступних для замовлення в ВНЗ](#_heading=h.3znysh7)



Рис. 1 – Загальний вигляд проїзного.

Правила користування студентським проїзним:

* За карткою дозволяється прохід тільки через автоматичний контрольний пункт після прикладання її до терміналу та появи зеленого світла на передній панелі.
* Картку не можна механічно навантажувати і тримати у потужному електромагнітному полі. Рекомендується носити її у захисному неметалевому футлярі або гаманці.
* Студентська електронна картка дійсна при наявності у її власника студентського квитка денної форми навчання.

До типових видів проїзних (Рис.1), які студент може замовити у ВНЗ відносяться:

* Метро
* Метро + Тролейбус
* Метро + Автобус
* Метро + Трамвай.

Також в кожної карточки проїзного існує ліміт поїздок, які власник може здійснити впродовж місяця.

Ліміти існують таких видів:

* 46 поїздок
* 62 поїздки
* Без обмеження кількості поїздок

Кожен проїзний має термін придатності 1 місяць, який зазначено на тильній стороні карти. Від типу проїзного залежить його вартість, також вона залежить від обраного ліміту.

З стандартними цінами на проїзні ви можете ознайомитись на табл. 1.1.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Тип карти  Ліміт | Метро | Метро- Автобус | Метро -Тролейбус | Метро - Трамвай |
| 46 | 160грн | 300грн | 300грн | 300грн |
| 62 | 210грн | 350грн | 350грн | 350грн |
| Без Обмеження | 320грн | 445грн | 445грн | 445грн |

Табл. 1.1. Політика ціноутворення проїзних

## 1.2 Основні положення та правила університету для щодо замовлення пільгових проїзних документів.

Задля оптимізації процесу видачі та замовлення проїзних університет запровадив низку правил:

* Замовляти проїзні можливо до 10-го числа поточного місяця
* Замовлення відбувається на наступний місяць
* Видачу проїзних організовує член профспілки
* На термін зимових канікул, замовлення проїзних на Січень та Лютий відбувається у Грудні.

Ці правила потрібно і врахувати при створенні додатку, оскільки вони прив’язані до механізму роботи Київпастранс.

З схемою процесу придбання проїзних ви можете ознайомитись на рис. 2.

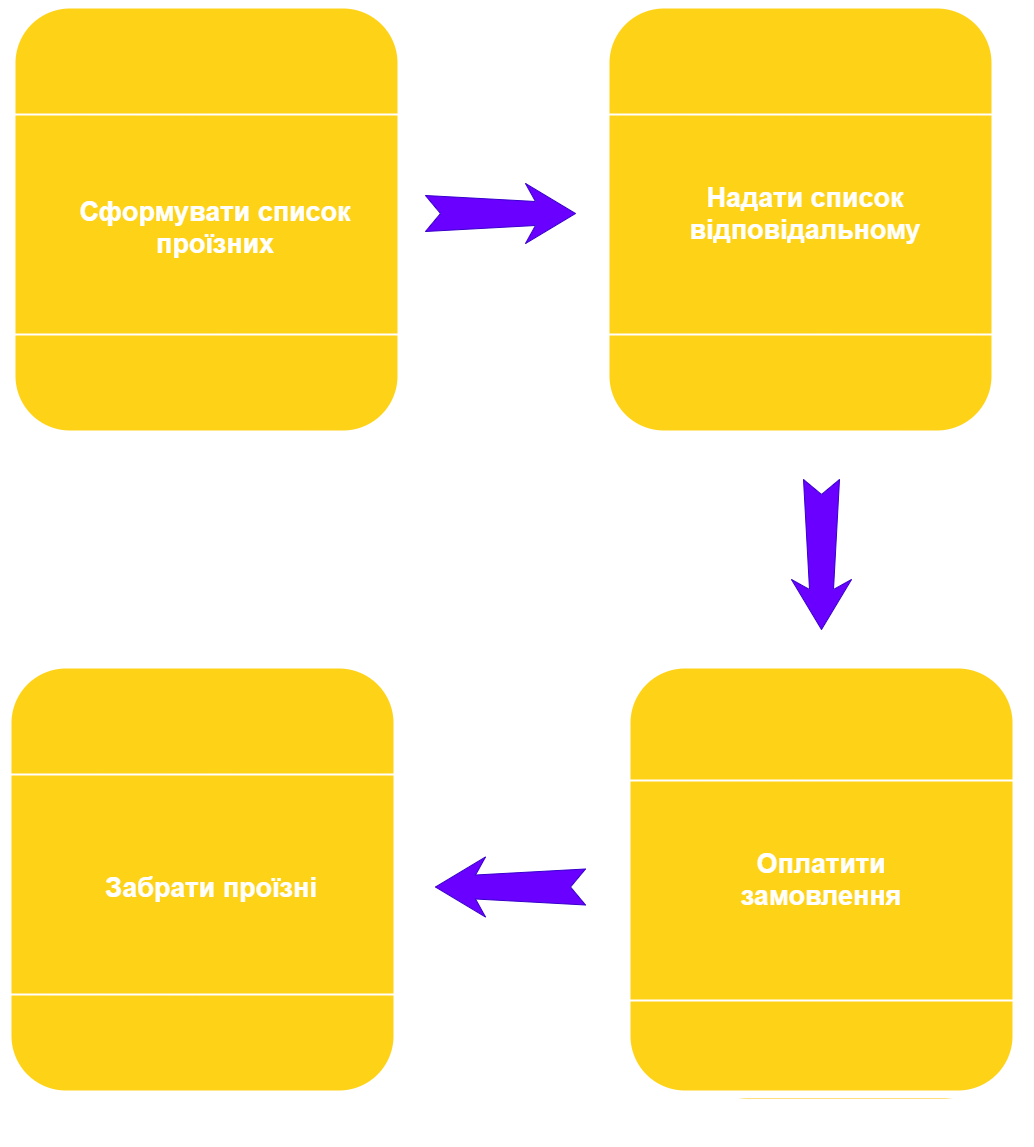


Рис. 2 – Послідовність процесу отримання проїзних

Дана схема описує процес взаємодії студента з відповідальною за проїзні особою.

Ми чітко спостерігаємо, що даний процес є неоптимальним, оскільки вимагає постійної взаємодії адміністратора з великою кількістю студентів.

Що , дуже часто, буває майже нереально справитись з фізичним навантаженням.

За допомогою сучасних інформаційних технологій можливо зменшити частину взаємодії до 2 пунктів зі сторони адміністратора:

* Підтвердити замовлення
* Видати проїзні

Що значно скоротить час, необхідний для обробки замовлень.

## 1.3 Огляд існуючих варіантів замовлення проїзних квитків ВНЗ

Студенти Фінансово-правового коледжу отримують проїзні через довірену особу тобто студенти передають кошти та замовлення через членів профспілок довіреній особі яка веде перелік всіх замовлень та в кінці місяця здійснює видачу проїзних (рис. 3).

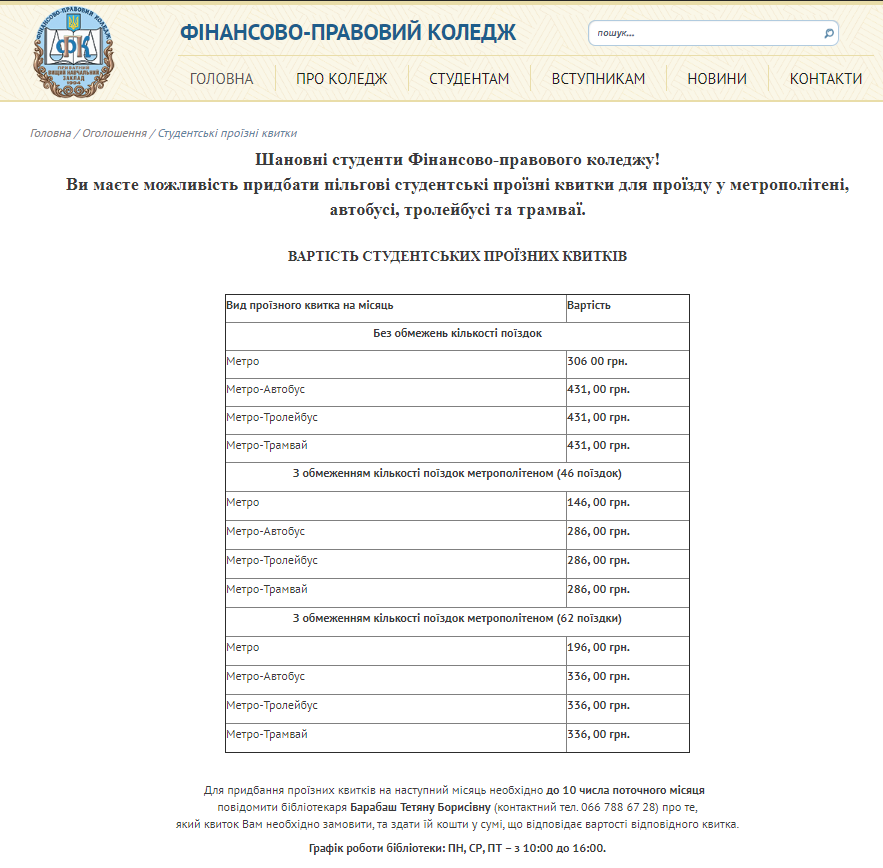


Рис. 3 – Сторінка замовлення Студентських проїзних квитків

Такий спосіб замовлення проїзних незручний тому що потрібно знайти відповідальну особу, якої може не бути на місці. Відповідальному за проїзні квитки самому потрібно вести перелік всіх замовлень та сповіщати студентів про наявність проїзних.

Студенти Національного медичного університету замовляють проїзні за допомогою банківського переказу (рис. 3). Тобто на рахунок відповідальної особи приходять банківські переводи с замовленнями.

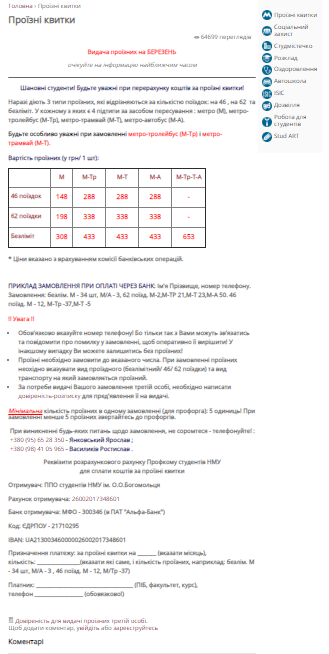


Рис. 4 – Сторінка замовлення студентських проїзних квитків в НМУ

Таким способом замовляти квитки зручніше студенти, тому що замовленням відправляють кошти за проїзні та саме текст замовлення через банківський рахунок. Такий спосіб має недоліки, тому що студенти важко відслідковувати замовлення також при оформлені замовлення дуже легко допустити помилку.

# [Розробка та опис функціонування додатку Proizd](#_heading=h.1t3h5sf)

## **2.1 Розробка та специфікація вимог до програмного додатку** Proizd

Розробка вимог є одним із найважливіших етапів створення програмного додатку. Визначення та подальша специфікація вимог дозволяє підвищити якість програмного продукту, знизити кількість його доробок та підвищити продуктивність [4].

Одним із засобів специфікації користувальницьких вимог є представлення їх за допомогою діаграм варіантів використання (діаграма прецедентів, Use Case Diagram), які дозволяють продемонструвати ключові можливості програми, доступні певним користувачам [5]. В результаті аналізу первинної інформації про особливості замовлення проїзних в Державному університеті телекомунікацій та на підставі даних про замовлення проїзних в ВНЗ (як в ДУТ, так і в інших навчальних закладах) було створено діаграму прецедентів додатку Proizd.

Діаграма наведена на рис. 5.

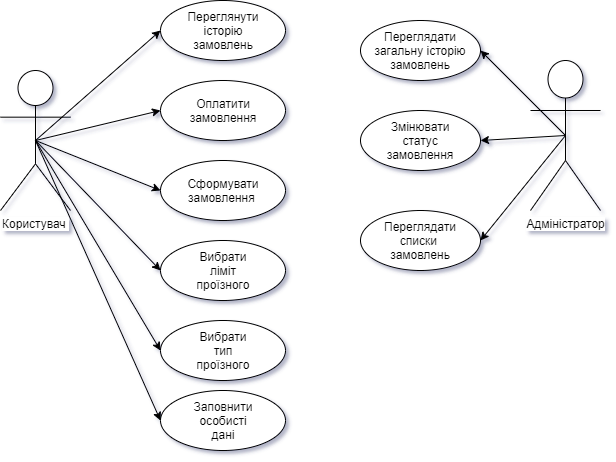


Рис. 5 – Діаграма прецедентів додатку Sked

В системі виділено два основних актори, які представлені користувачами «Користувач» та «Адміністратор».

Для кожного з акторів в додатку передбачені власні прецеденти. Актор «Студент» має доступ до наступних функцій:

* «Заповнити особисті дані» – прецидент, необхідний для коректної простежуваності, оплати замовлення та його видачі.
* «Вибрати тип проїзного» – для його вибору необхідно знати шифри типів та визначити який конкретно потрібен вам.
* «Вибрати ліміт проїзного» – є доповнюючим до «Вибрати тип проїзного», щоб користувач мав можливість обрати карту з потрібним лімітом.
* «Сформувати замовлення» – як вже зрозуміло, користувач може замовляти не 1 проїзний, а декілька, наприклад своїм одногрупникам, тому замовлення може містити декілька типів карт.
* «Оплатити замовлення» – користувач повинен прикріпити чек про оплату замовлення не пізніше 10-го числа поточного місяця. Лише тоді він зможе успішно отримати свої проїзні.
* «Переглядати історію замовлень» – у користувача є доступ до історії своїх замовлень, їх статусів та повного переліку проїзних, що туди входять.

Актор «Адміністратор» має доступ до наступних функцій:

* «Переглядати списки замовлень» – адміністратор має доступ до всіх замовлень та їх параметрів, таких як статус, перелік проїзних та дані про замовника.
* «Змінювати статус замовлення» – адміністратор повинен перевіряти чи користувач сплатив замовлення, та відносно цього змінювати його статус.
* «Переглядати загальну історію замовлень» – адміністратор має доступ до всієї історії замовлень, для підрахунків статистичних даних та звітності.

Перелік вимог, які повинні бути реалізовані без прив’язання до користувачів, а виконуватися системною частиною додатку можна сформулювати наступним чином:

* Взаємодія між клієнтами та адміністратором повинна відбуватись за допомогою Google Sheets таблички
* Контроль та запис інформації про замовлення відбувається за допомогою веб сервера, який забезпечує взаємодію між адміністратором та звичайними користувачами.
* Обмеження термінів прийому та оплати замовлень, згідно до п. 1.2

## 2.2 Опис засобів програмної реалізації

Оскільки цільовою аудиторією користувачів системи є люди, що володіють мобільними пристроями на базі операційної системи Android [6], то було обрано відповідні засоби реалізації додатку Proizd.

**2.2.1 Реалізація клієнтської частини**

javaScript (JS) — динамічна, [об'єктно-орієнтована](https://uk.wikipedia.org/wiki/%D0%9E%D0%B1%27%D1%94%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D1%96%D1%94%D0%BD%D1%82%D0%BE%D0%B2%D0%B0%D0%BD%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F) [прототипна](https://uk.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%82%D0%BE%D1%82%D0%B8%D0%BF%D0%BD%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F) [мова програмування](https://uk.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B2%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F). Найчастіше використовується для створення сценаріїв [веб-сторінок](https://uk.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D1%81%D1%82%D0%BE%D1%80%D1%96%D0%BD%D0%BA%D0%B0), що надає можливість на стороні [клієнта](https://uk.wikipedia.org/wiki/%D0%9A%D0%BB%D1%96%D1%94%D0%BD%D1%82-%D1%81%D0%B5%D1%80%D0%B2%D0%B5%D1%80%D0%BD%D0%B0_%D0%B0%D1%80%D1%85%D1%96%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B0) (пристрої кінцевого користувача) взаємодіяти з користувачем, керувати браузером, [асинхронно](https://uk.wikipedia.org/wiki/AJAX) обмінюватися даними з [сервером](https://uk.wikipedia.org/wiki/%D0%9A%D0%BB%D1%96%D1%94%D0%BD%D1%82-%D1%81%D0%B5%D1%80%D0%B2%D0%B5%D1%80%D0%BD%D0%B0_%D0%B0%D1%80%D1%85%D1%96%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B0), змінювати [структуру](https://uk.wikipedia.org/wiki/%D0%9E%D0%B1%27%D1%94%D0%BA%D1%82%D0%BD%D0%B0_%D0%BC%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C_%D0%B4%D0%BE%D0%BA%D1%83%D0%BC%D0%B5%D0%BD%D1%82%D0%B0) та [зовнішній вигляд](https://uk.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D0%B4%D0%B8%D0%B7%D0%B0%D0%B9%D0%BD) [веб-сторінки](https://uk.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D1%81%D1%82%D0%BE%D1%80%D1%96%D0%BD%D0%BA%D0%B0).

JavaScript класифікують як [прототипну](https://uk.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%82%D0%BE%D1%82%D0%B8%D0%BF%D0%BD%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F) (підмножина [об'єктно-орієнтованої](https://uk.wikipedia.org/wiki/%D0%9E%D0%B1%27%D1%94%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D1%96%D1%94%D0%BD%D1%82%D0%BE%D0%B2%D0%B0%D0%BD%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F)), [скриптову мову](https://uk.wikipedia.org/wiki/%D0%A1%D0%BA%D1%80%D0%B8%D0%BF%D1%82%D0%BE%D0%B2%D0%B0_%D0%BC%D0%BE%D0%B2%D0%B0) програмування з динамічною типізацією. Окрім прототипної, JavaScript також частково підтримує інші парадигми програмування ([імперативну](https://uk.wikipedia.org/wiki/%D0%86%D0%BC%D0%BF%D0%B5%D1%80%D0%B0%D1%82%D0%B8%D0%B2%D0%BD%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F) та частково [функціональну](https://uk.wikipedia.org/wiki/%D0%A4%D1%83%D0%BD%D0%BA%D1%86%D1%96%D0%BE%D0%BD%D0%B0%D0%BB%D1%8C%D0%BD%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F)) і деякі відповідні архітектурні властивості, зокрема: [динамічна](https://uk.wikipedia.org/wiki/%D0%94%D0%B8%D0%BD%D0%B0%D0%BC%D1%96%D1%87%D0%BD%D0%B0_%D1%82%D0%B8%D0%BF%D1%96%D0%B7%D0%B0%D1%86%D1%96%D1%8F) та слабка [типізація](https://uk.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D1%82%D0%B8%D0%BF%D1%96%D0%B7%D0%B0%D1%86%D1%96%D1%97), автоматичне керування пам'яттю, прототипне [наслідування](https://uk.wikipedia.org/wiki/%D0%9D%D0%B0%D1%81%D0%BB%D1%96%D0%B4%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F)), [функції](https://uk.wikipedia.org/wiki/%D0%A4%D1%83%D0%BD%D0%BA%D1%86%D1%96%D1%8F_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F)) як [об'єкти першого класу](https://uk.wikipedia.org/wiki/%D0%9E%D0%B1%27%D1%94%D0%BA%D1%82_%D0%BF%D0%B5%D1%80%D1%88%D0%BE%D0%B3%D0%BE_%D0%BA%D0%BB%D0%B0%D1%81%D1%83).

Мова JavaScript використовується для:

* написання сценаріїв [веб-сторінок](https://uk.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D1%81%D1%82%D0%BE%D1%80%D1%96%D0%BD%D0%BA%D0%B0) для надання їм інтерактивності;
* створення односторінкових [веб-застосунків](https://uk.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D0%B7%D0%B0%D1%81%D1%82%D0%BE%D1%81%D1%83%D0%BD%D0%BE%D0%BA) ([React](https://uk.wikipedia.org/wiki/React), [AngularJS](https://uk.wikipedia.org/wiki/AngularJS), [Vue.js](https://uk.wikipedia.org/wiki/Vue.js));
* [програмування](https://uk.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F) на стороні сервера ([Node.js](https://uk.wikipedia.org/wiki/Node.js));
* стаціонарних [застосунків](https://uk.wikipedia.org/wiki/%D0%97%D0%B0%D1%81%D1%82%D0%BE%D1%81%D1%83%D0%BD%D0%BE%D0%BA) ([Electron](https://electron.atom.io/), [NW.js](https://nwjs.io/));
* мобільних застосунків ([React Native](https://facebook.github.io/react-native/), [Cordova](https://cordova.apache.org/));

Для зручної реалізації клієнтської частини було використано фреймворк React Native.

React Native представлений компанією Facebook у 2015 році, що застосовує React архітектуру до нативних IOS, Android та UWP додатків

Принципи роботи React Native в основному такі ж, як ReactJS, за винятком того, що він не маніпулює DOM через VirtualDom.

Він працює у фоновому процесі (який інтерпретує Javascript код написаний розробниками) безпосередньо на кінцевому пристрої і спілкується з нативною платформою. Очевидно, що Facebook виправив помилку, про яку Марк Цукерберг згадував в 2012 році. React Native взагалі не покладається на HTML, все написано на Javascript і залежить від нативних SDK.

React не намагається надати повну "схему додатків". Він безпосередньо спрямований на побудову користувацьких інтерфейсів, і тому не включає в себе безліч інструментів, які деякі розробники вважають необхідними для створення програми. Це дозволяє вибрати будь-які бібліотеки, які розробник вважає за краще виконувати, щоб виконати певних завдань, таких як здійснення доступу до мережі або локальне зберігання даних.

Для підтримки концепції React щодо одностороннього потоку даних, архітектура Flux являє собою альтернативу популярній архітектурі Model-view-controller. Flux – це дії, які надсилаються через центральний диспетчер до сховища, а зміни з сховища надсилають назад до перегляду. У використанні разом з React, це поширення здійснюється через компоненти властивостей.

**2.2.2 Реалізація серверної частини**

C# — об'єктно-орієнтована мова програмування з безпечною системою типізації для платформи .NET. Розроблена Андерсом Гейлсбергом, Скотом Вілтамутом та Пітером Гольде під егідою Microsoft Research (при фірмі Microsoft).

Синтаксис C# близький до С++ і Java. Мова має строгу статичну типізацію, підтримує поліморфізм, перевантаження операторів, вказівники на функції-члени класів, атрибути, події, властивості, винятки, коментарі у форматі XML. Перейнявши багато що від своїх попередників — мов С++, Object Pascal, Модула і Smalltalk — С#, спираючись на практику їхнього використання, виключає деякі моделі, що зарекомендували себе як проблематичні при розробці програмних систем, наприклад множинне спадкування класів (на відміну від C++).

ASP.NET Core — вільне та відкрите програмне забезпечення каркасу веб застосунків, з продуктивністю вищою ніж у ASP.NET, розроблена корпорацією Microsoft і співтовариством. Це модульна структура, яка працює як на повній платформі .NET Framework, так і на платформі .NET Core.

Фреймворк являє собою повний перепис, який об'єднує раніше окремі ASP.NET MVC та ASP.NET Web API у єдину програмувальну модель.

Не зважаючи на те, що це є новим фреймворком, побудованим на новому веб-стеку, ASP.NET Core має високу ступінь сумісності концепцій з ASP.NET MVC, який об'єднує функціональність MVC, Web API та Web Pages. В попередніх версіях платформи дані технології реалізовані окремо і тому містять багато дублювальної функціональності. Тепер це об'єднано в одну програмну модель ASP.NET Core MVC. Веб-форми повністю вийшли в минуле. Програми ASP.NET Core підтримують програмні версії, в якій різні програми, що працюють на одному комп'ютері, можуть орієнтуватися на різні версії ASP.NET Core. Це не можливо з попередніми версіями ASP.NET Core.

## 2.3 [Архітектура додатку](#_heading=h.17dp8vu)

Для забезпечення незалежності логіки додатку від його інтерфейсу в розробці було використано модель клієнт-серверної архітектури тобто додаток має дві незалежні частини клієнтську частину на телефоні та веб сервер.

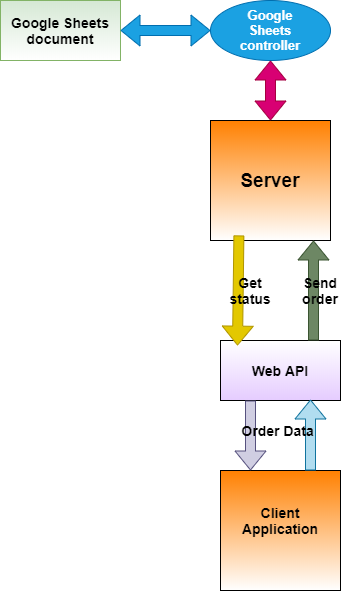


Рис 6. [Архітектура](#_heading=h.17dp8vu) серверної частини додатку

**2.3.1 Серверна частина додатку**

Для написання серверної частини було обрано С# а для зберігання даних про замовлення Google Sheets.

Дані в таблиці розміщені наступним чином: лист с даними про проїзні та їх ціни та листи з даними про замовлення для кожного місяця. В листі з даними про замовлення зберігатиметься інформація про замовника, дані самого замовлення та статус замовлення також в цій на цьому листі розміщена інформація про загальну кількість замовлених проїзних.

Сервер буде обробляти запити від додатку на телефоні. Сервер може обробити декілька типів запиту:

* Створення нового замовлення;
* Запит інформації про проїзні;
* Запит інформації про статус замовлення.

Після надходження запиту на сервер, він обробляє запит, зберігає дані про замовлення у таблицю або відправляє необхідну інформацію до клієнтської частини.. При необхідності (кожного місяця) сервер буде створювати нову сторінку в документі зі списком замовлень проїзних на наступний місяць.

**2.3.2 Клієнт - серверна взаємодія.**

Додаток на телефоні та сервер повинні якимось чином обмінюватися інформацією для цих цілей було обрано архітектуру мережевих протоколів REST.

**2.3.3 Архітектура клієнтської частини**

Flux-архітектура - архітектурний підхід або набір шаблонів програмування для побудови призначеного для користувача інтерфейсу веб-додатків, що поєднується з реактивним програмуванням і побудований на односпрямованих потоках даних.

Згідно із задумом творців і незважаючи на те, що Facebook надав реалізацію Flux на додаток до React, Flux не є ще одним веб-фреймворком, а є архітектурним рішенням.

Основною відмінною рисою Flux є одностороння спрямованість передачі даних між компонентами Flux-архітектури. Архітектура накладає обмеження на потік даних, зокрема, виключаючи можливість поновлення стану компонентів самими собою. Такий підхід робить потік даних передбачуваним і дозволяє легше простежити причини можливих помилок в програмному забезпеченні .

У мінімальному варіанті Flux-архітектура може містити три шари, які взаємодіють один по одному :

* Actions (дії)
* Stores (сховища)
* Views (подання)

Хоча зазвичай між діями і сховищами додають Dispatcher (диспетчер).

В першу чергу Flux працює з інформаційною архітектурою, яка потім відбивається в архітектурі програмного забезпечення, тому рівень уявлень слабо зачеплений з іншими рівнями системи .

Дії (англ. Actions) - вираз подій (часто для дій використовуються просто імена - рядки, що містять деякий «дієслово»). Диспетчери передають дії нижчого компонентів (сховищ) по одному. Нова дія не передається поки попереднє повністю не оброблено компонентами. Дії через роботу джерела дії, наприклад, користувача, надходять асинхронно, але їх диспетчеризація є синхронним процесом . Крім імені (англ. Name), дії можуть мати корисне навантаження (англ. Payload), що містить пов'язані з дією дані .

Диспетчер (англ. Dispatcher) призначений для передачі дій сховищ. У спрощеному варіанті диспетчер може взагалі не виділятися, як єдиний на весь додаток. У диспетчері сховища реєструють свої функції зворотного виклику (callback) і залежності між сховищами.

Сховище (англ. Store) є місцем, де зосереджено стан (англ. State) додатку. Інші компоненти, згідно Flux, не мають значного (з точки зору архітектури) стану. Зміна стану сховища відбувається строго на основі даних дії і старого стану сховища .

Подання (англ. View) - компонент, звичайно відповідає за видачу інформації користувачеві. У Flux-архітектурі, яка може технічно не торкатися внутрішнього облаштування уявлень взагалі, це - кінцева точка потоків даних. Для інформаційної архітектури важливо тільки, що дані потрапляють в систему (тобто, назад в сховища) тільки через дії .

Основні відмінні риси:

Синхронність: всі методи зворотного виклику, зареєстровані для кожної дії, синхронні у виконанні, саме ж дія може викликатися джерелом асинхронно.

Інверсія управління: потік управління передається відповідному сховища і цільової функції зворотного виклику.

Семантичні дії: дія, що викликається джерелом, містить смислове інформацію, що дозволяє відповідному сховища вибрати правильний метод виконання.

Відсутність каскадів дій: Flux забороняє каскадні (вкладені) дії.

порівняння

У порівнянні з широко використовуваному в тому числі в веб-програмуванні шаблоном видавець-передплатник, Flux пропонує статичну структуру передачі повідомлень, при якій повідомлення отримує кожен компонент. Компонент вирішує, що робити з цим повідомленням. Це дозволяє обійти деякі архітектурні проблеми видавця-передплатника, пов'язані з порядком оповіщення компонентів при додаванні нових компонентів (проблеми масштабування), а також з додатковою складністю, пов'язаною з підпискою і відмовою від підписки протягом життєвого циклу компонентів, при якому можлива втрата значущих для компонента повідомлень .

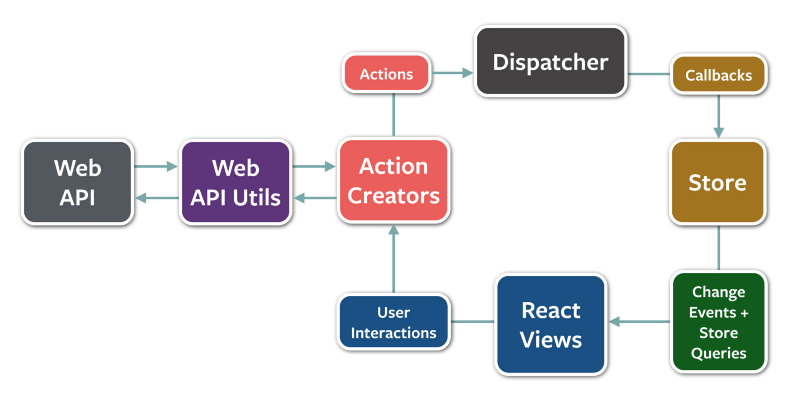


Рис. 7 – Архітектура клієнтської частини додатку

Presentation-шар відповідає за UI та взаємодію з користувачем (обробка натискань на кнопки та ін.). Шар складається з об'єктів presenter та view. Шар View містить логіку відображеня, тобто view знає як потрібно відобразити дані. Шар Presenter - відповідає за те, що потрібно відобразити. Data шар відповідає за роботу з даними, а саме їх зберігання та надання. В цьому шарі виконується завантажування інформації про студента/викладача, завантаження розкладу та зберігання його у базу даних. Цей шар складається з об'єктів repository. Domain шар відповідає за логіку додатку. Шар представлено об'єктами Interactor, які є містком між presentation та data шарами.

## 2.4 Опис функціонування додатку

Додаток Proizd має 4 екрани:

* Order,
* Cart,
* History,
* About.

«Order» (рис. 8) – перший екран, який бачить користувач при запуску додатку. На цьому екрані потрібно заповнити особисту інформацію користувача: ПІБ, група, номер телефону.

Також на цьому екрані користувач може додавати проїзні у свою корзину.

Вибір типу проїзного та його ліміту організований за допомогою випадних списків, це дозволяє користувачу швидше вводити інформацію та значно зменшує можливість помилкового введення.

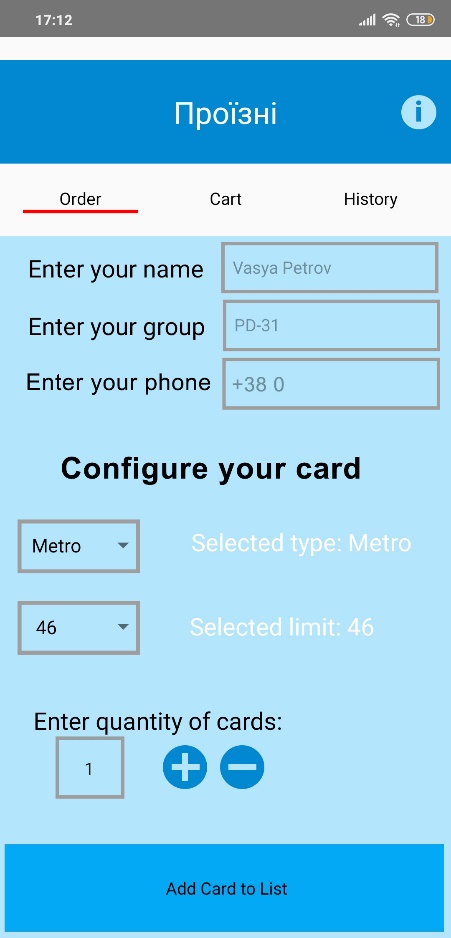
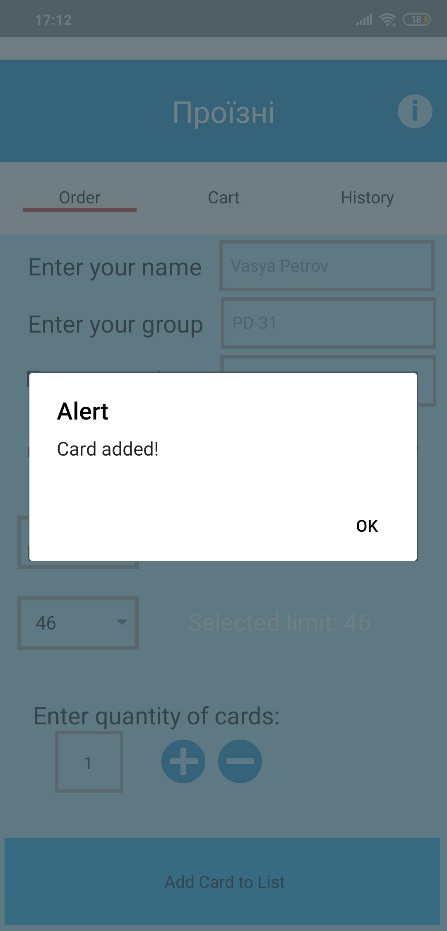


Рис. 8 – Екрани введення інформації та вибору проїзного

По кліці на кнопку «Add Card to List» сформоване замовлення потрапляє у корзину(Cart)

«Cart»(рис. 9) – екран, на якому користувач може переглянути список проїзних, їх кількість та тип.

На даному екрані також можна видалити непотрібні проїзні з корзини та підтвердити замовлення.

При кліці на кнопку «Order your cards » вся інформація про замовлення відправляється на сервер, який відповідно записує її в табличку.

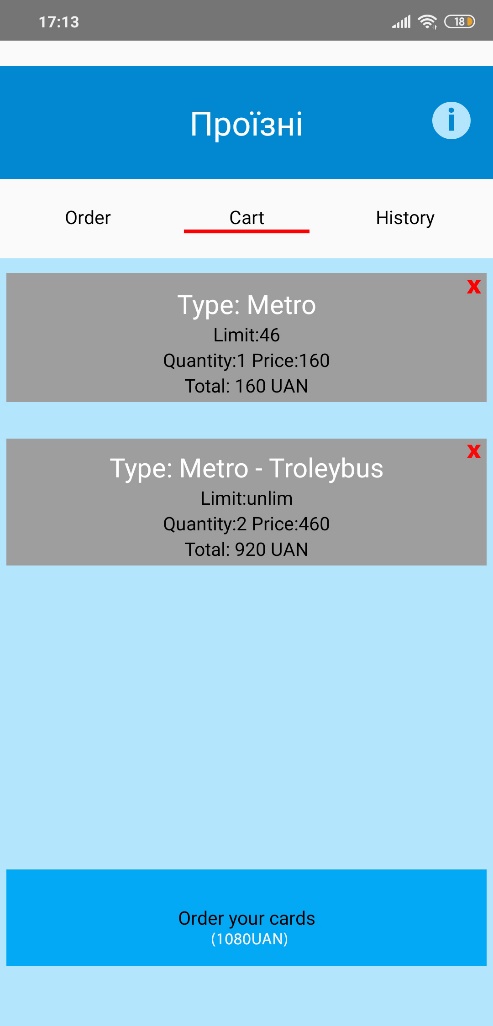


Рис. 9 – Екран корзини

Екран «History»(рис. 10) призначено для історії замовлень. Також одна з головних його функцій, це відправка чека про оплату замовлення, який буде підтверджувати ваше замовлення, та змінювати статус з «чекає підтвердження» в «підтверджено».

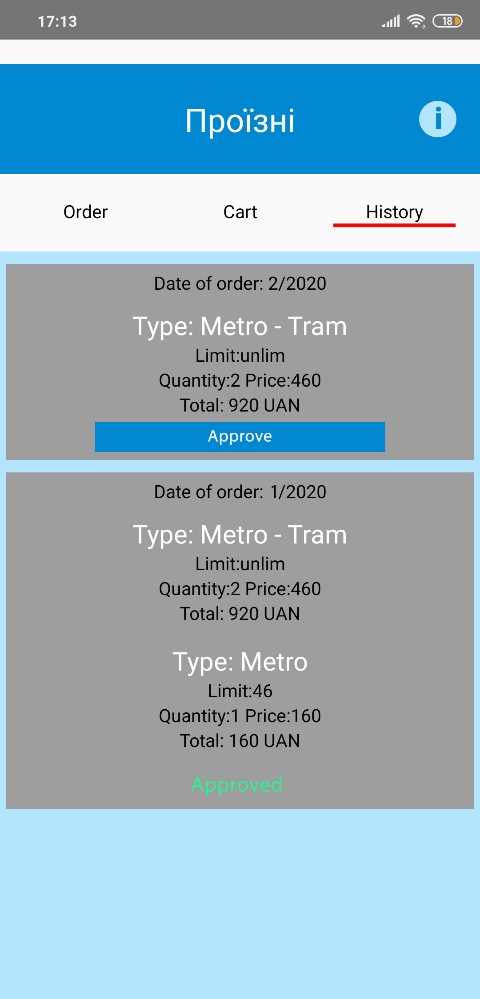


Рис. 10 – Екран історії замовлень

Екран About містить загальну інформацію про додаток: дані про автора та ліцензії програмного забезпечення, яке було використано при створенні додатку. Вигляд екрану About наведено на рис. 11.

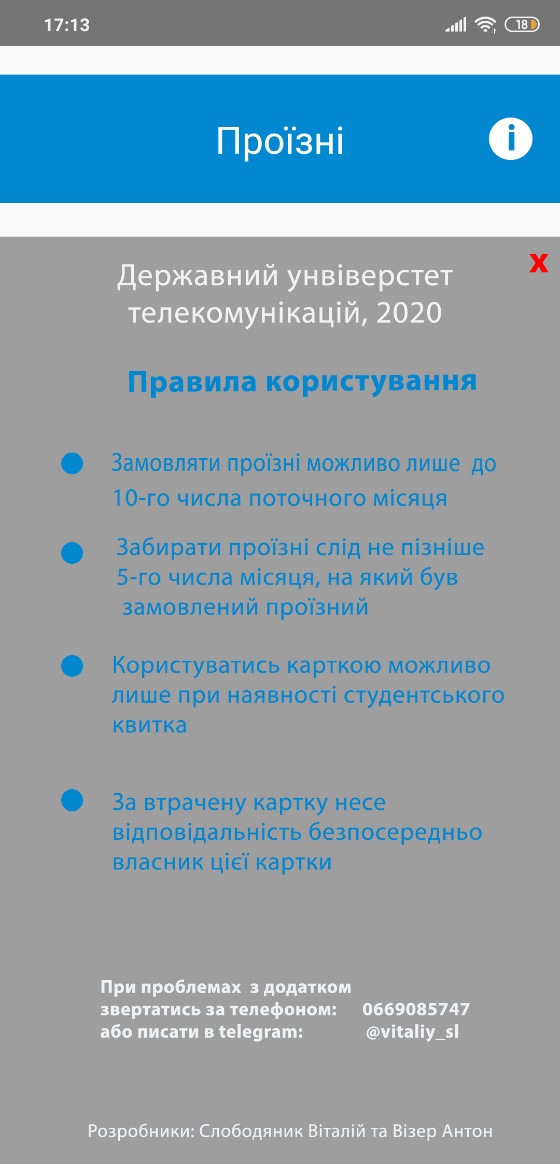


Рис. 11 – Екран About

## Висновки

Проведено аналіз процесу замовлення проїзних у вищих навчальних закладах, виявлено особливості даного процесу в Державному університеті телекомунікацій.

Зроблено огляд та аналіз програмних засобів, які використовуються для зручного формування списків проїзних в інших ВНЗ. Виявлено функціональні можливості та недоліки існуючих програм, що представляють розклади ВНЗ.

Розроблено вимоги до мобільного додатку Proizd для автоматизації збору списків проїзних та їх оплати в Державному університеті телекомунікацій.

Проведено специфікацію вимог із використанням діаграми прецедентів UML.

Проведено аналіз сучасних програмних засобів реалізації мобільних додатків на базі платформи Android та описано бібліотеки, шаблони та фреймворки, використані в додатку Proizd.

Розроблена архітектура додатку із використанням патерну Flux, який дозволив оптимізувати роботу додатку, відділивши інтерфейсні рішення від логіки обробки даних.

Розроблено та описано програмний додаток Proizd, який дозволяє швидко сформувати замовлення проїзних у зручній формі на мобільному пристрої .

Напрямки подальших досліджень полягають у наступному:

* інтеграція з системами онлайн оплати для більшої зручності оформлення;
* оптимізація представлення додатку для різних форм-факторів мобільних пристроїв;
* введення персонального кабінету користувача.

## Література

1. Максимов, Н.В. Современные информационные технологии: Учебное пособие / Н.В. Максимов, Т.Л. Партыка, И.И. Попов. – М.: Форум, 2013. – 512 c.
2. Кронин Д. Алан Купер об интерфейсе. Основы проектирования взаимодействия / Д. Кронин, А. Купер А. Р. Рейман. – СПб.:
3. Вигерс К., Битти Д. Разработка требований к программному обеспечению. – 3-е изд., доп. — Пер. с англ. — М.: Русская редакция; СПб.: БХВ-Петербург, 2014. — 736 стр.: ил.
4. Ларман, Крэг. Применение UML и шаблонов проектирования.: Пер. с англ..: Уч. Пос. – М.: Издательский дом «Вильямс», 2001. – 496 с.: ил. – Парал. Тит. Англ.
5. Майер P. Android 2: программирование приложений для планшетных компьютеров и смартфонов / Р. Майер. – М. : Эксмо, 2011. – 672 с.
6. Жемеров Д. Kotlin в действии./ С. Исакова Д. Жемеров. - М: ДМК Пресс, 2017. – 402 с.
7. TechYourChance. Professional Android Development Blog. MVP and MVC Architectures in Android – part 1. https://www.techyourchance.com/mvp-mvcandroid-1/