Державний університет Телекомунікацій

Спеціальність: Інженерія програмного забезпечення

**Семестрова робота**

**Шифр роботи «Proizd»**

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

Автори роботи:

студенти групи ПД-31

Слободяник Віталій та Візер Антон

2019-2020 навчальний рік

**Зміст**

[Вступ 3](#_Toc36045577)

[1. Огляд та аналіз особливостей замовлення проїзних у ВНЗ 4](#_Toc36045578)

[1.1 Аналіз типів проїзних карт, доступних для замовлення в ВНЗ 4](#_Toc36045579)

[1.2 Основні положення та правила університету щодо замовлення пільгових проїзних документів. 5](#_Toc36045580)

[2. Розробка та опис функціонування додатку Proizd 7](#_Toc36045581)

[2.1 Розробка та специфікація вимог до програмного додатку Proizd 7](#_Toc36045582)

[2.2 Опис засобів програмної реалізації 9](#_Toc36045583)

[2.3 Архітектура додатку 11](#_Toc36045584)

[2.4 Опис функціонування додатку 17](#_Toc36045585)

[3. Ролі та менеджмент проекту 21](#_Toc36045586)

[3.1 Підхід до розробки 21](#_Toc36045587)

[3.2 Ролі на проекті 21](#_Toc36045588)

[Висновки 22](#_Toc36045589)

# 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 дорогим, існують студентські типи проїзних карт, ціна однієї поїздки в яких від 40% до 50% дешевша, в порівнянні зі звичайними.

Згідно з правилами Київпастранс, 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

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. Це дозволить не тільки витрачати менше часу на весь процес, а й зробить його більш відкритим та зручним.

# Огляд та аналіз особливостей замовлення проїзних у ВНЗ

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



Рис. 1 – Загальний вигляд проїзного.

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.

Зі стандартними цінами на проїзні (2020р.) 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 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.

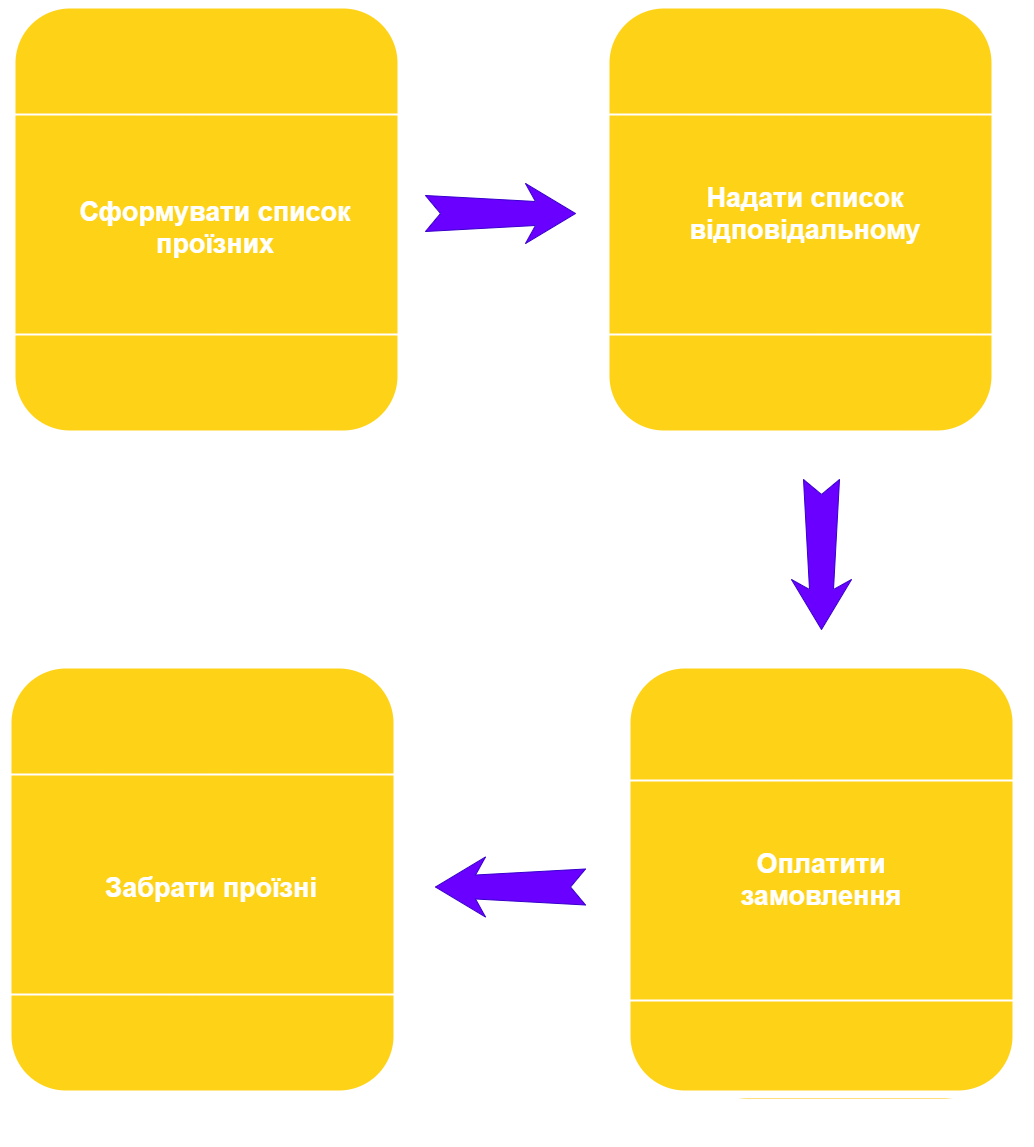


Рис. 2 – Послідовність процесу отримання проїзних

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.

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

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

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. 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.

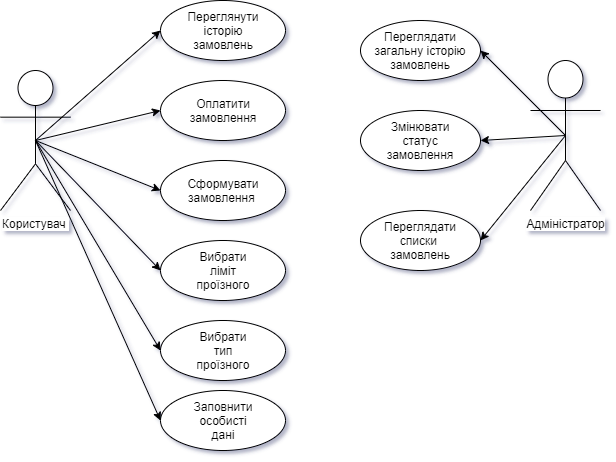


Рис. 5 – Діаграма прецедентів додатку Proizd

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).

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).

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.

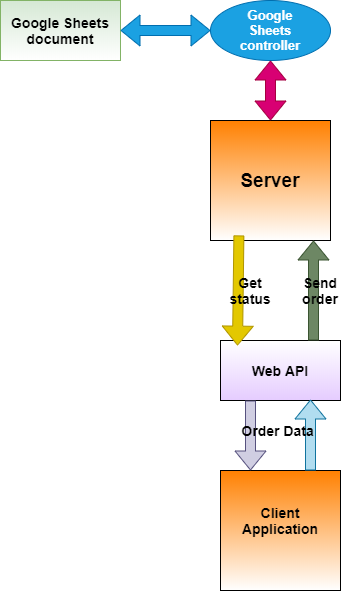


Рис 6. [Архітектура](#_heading=h.17dp8vu) серверної частини додатку

**2.3.1 Серверна частина додатку**

Для написання серверної частини було обрано мову програмууання С# ,а для зберігання даних про замовлення таблицю Google Sheets та Google Sheets API для їхньої взаємодії.

Дані в таблиці розміщені наступним чином:

* лист з даними про проїзні та їх ціни
* листи з даними про замовлення для кожного місяця.

В листі з даними про замовлення зберігатиметься інформація про замовника, дані самого замовлення та статус замовлення.

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 API.

Загалом це мережеві HTTP запити, 4 типів(в основниму), а саме

* GET
* PUT
* POST
* DELETE

Передача інформації відбуватиметься за допомогою JSON документів.

Наприклад запит до сервера щодо надходження повідомлення про нове замовлення виглядатиме так:

POST 192.168.1.3:5050/newOrder

Та тіло запиту:

{

“id”: “sd72hcsais12wj8”,

“date”: ”4/2020”,

“name”: “Vasya Petrov”,

… - вся необхідна інформація про замовника

“order”:[

{

“type”:”Metro-Bus”,

“limit”: “64”,

“quantity”: 2,

“price”: 320,

“totalPrice”: 640

},{

“type”:”Metro-Troleybus”,

“limit”: “unlim”,

“quantity”: 2,

“price”: 450,

“totalPrice”: 900

}]

}

Як бачите, інформація структурована, та однотипна, що дозволяє її легко обробляти.

**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 .

**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 смислову інформацію, що дозволяє відповідному сховищу 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 MVC, 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, 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 .

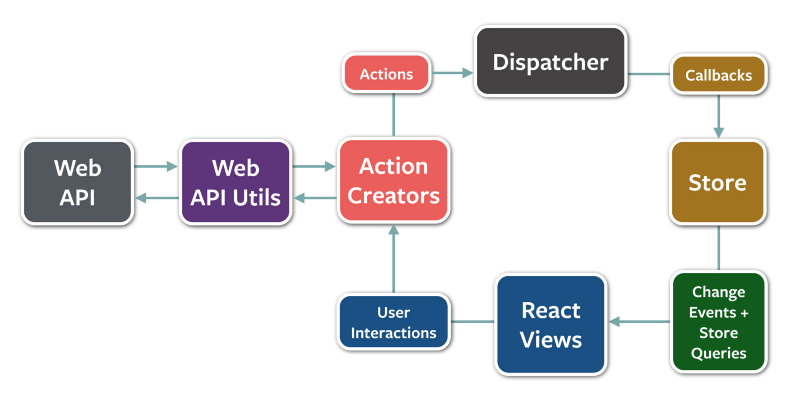


Рис. 7 – Архітектура клієнтської частини додатку

## 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.

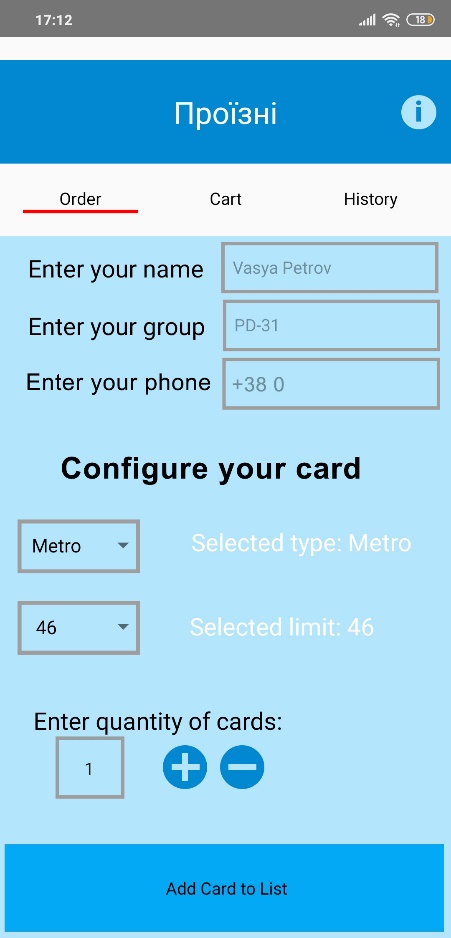
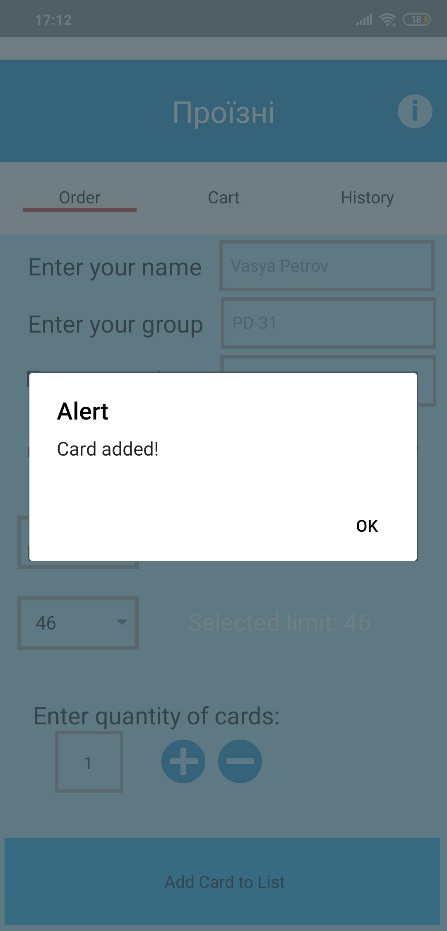


Рис. 8 – Екрани введення інформації та вибору проїзного

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.

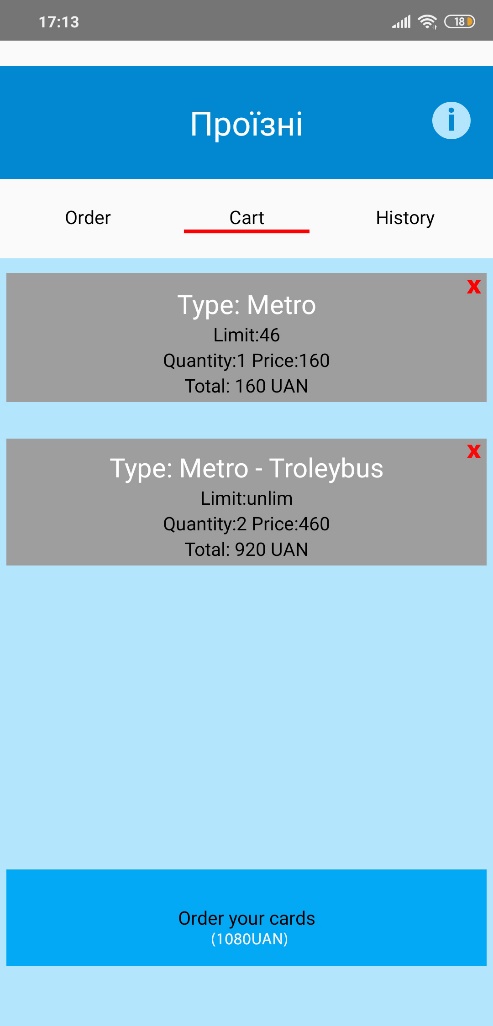


Рис. 9 – Екран корзини

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».

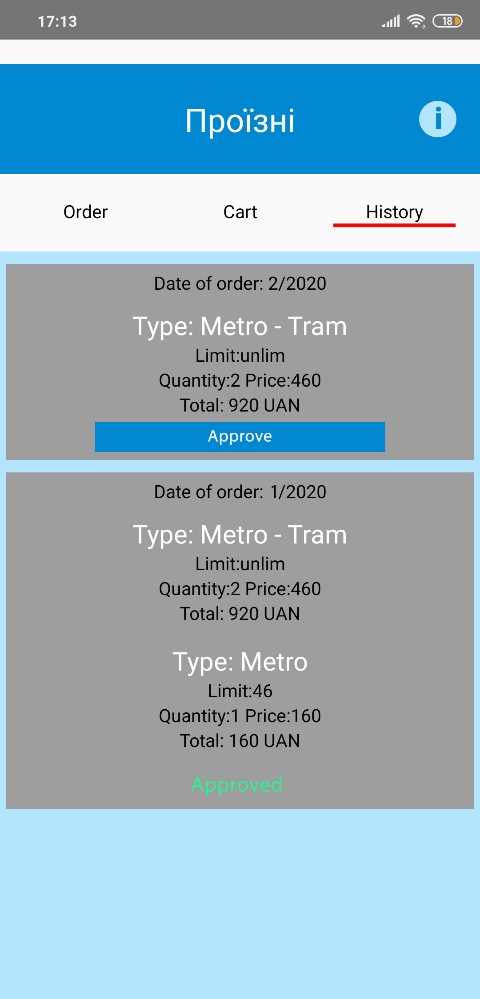


Рис. 10 – Екран історії замовлень

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.

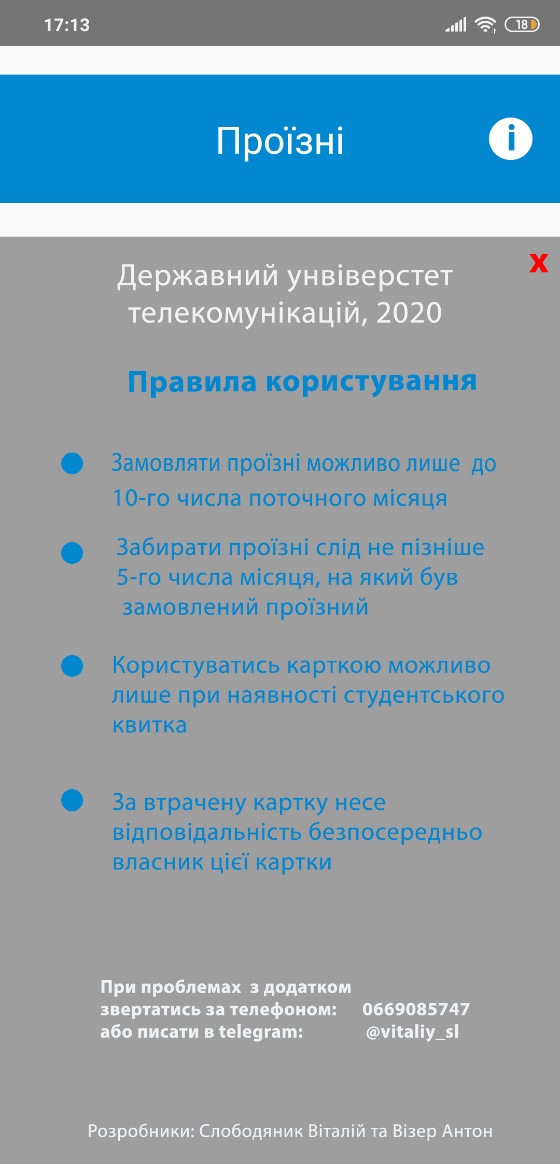


Рис. 11 – Екран About

# Ролі та менеджмент проекту

## 3.1 Підхід до розробки

Для збільшення ефективності розроби ми вирішили використати ітеративний підхід Scrum.

Сама розробка буде поділена на 2-тижневі спрінти, за які ми повинні реалізувати новий функціонал для додатку.

Product Backlog буде оформлений за допомогою сервісу Trello.

Посилання: <https://trello.com/b/ksJzymsC/proizd>

## 3.2 Ролі на проекті

|  |  |
| --- | --- |
| Учасник | Роль |
| 1.Візер Антоній | Backend Developer, Data Architect, PM |
| 2.Слободяник Віталій | Frontend Developer, System Architect, QA |

## 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.