Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»
Кафедра інформатики факультету інформатики

Розробка кабінету вступника в мобільному застосунку під iOS для цифрового університету

Текстова частина до кваліфікаційної роботи
за спеціальністю «Інженерія програмного забезпечення»

Керівник кваліфікаційної роботи
доктор технічних наук, професор,
Глибовець А. М.

_____
(підпис)
“___” _____ 2026 р.
Виконав студент Франків Я. О.
“___” _____ 2026 р.

Київ 2026

Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Кафедра інформатики факультету інформатики

ЗАТВЕРДЖУЮ

Зав. кафедри інформатики,

доцент, кандидат наук

_____ Нагірна А. М.

(підпис)

“____” _____ 2026 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

на кваліфікаційну роботу

студенту Франківу Ярославу Олександровичу 4-го курсу факультету інформатики

ТЕМА: Розробка кабінету вступника в мобільному застосунку під iOS для цифрового університету

Індивідуальне завдання

Анотація

Вступ

Розділ 1. Вступна кампанія в контексті цифрового університету

Розділ 2. Проєктування кабінету вступника

Розділ 3. Імплементація та тестування системи

Висновки

Список використаних джерел

Додатки

Дата видачі “___” _____ 2026 р.

Керівник _____ Завдання отримано _____

Календарний план виконання кваліфікаційної роботи

Тема: Розробка кабінету вступника в мобільному застосунку під iOS для цифрового університету

№ з/п	ПЕРЕЛІК РОБІТ	Термін виконання	Примітка
1	Отримання теми кваліфікаційної роботи	Листопад–грудень 2025	
2	Створення плану роботи	Кінець грудня 2025	
3	Ознайомлення з проєктом	Січень 2026	
4	Розроблення практичної частини проєкту	Лютий–березень 2026	
5	Написання першого розділу роботи	Початок квітня 2026	
6	Опис практичної частини роботи	Середина квітня 2026	
7	Перегляд змісту роботи з керівником	Кінець квітня 2026	
8	Внесення змін відповідно до зауважень наукового керівника	Кінець квітня 2026	
9	Створення презентації	Початок травня 2026	
10	Захист кваліфікаційної роботи	27.05.2026	

Студент Франків Я. О.

_____

Керівник Глибовець А. М.

_____

“_____” _____ 2026 р.

ЗМІСТ

АНОТАЦІЯ

У кваліфікаційній роботі розглянуто процес проєктування та розробки кабінету вступника в мобільному застосунку SmartUKMA для iOS. Метою розробки є створення зручного інструменту для вступника, що забезпечує доступ до персональних вступних даних, заяв, рейтингових позицій, статусів і строків підтвердження вибору місця навчання.

У ході реалізації було проаналізовано процес вступної кампанії, роль ЄДЕБО як джерела вступних даних та місце кабінету вступника в цифровій екосистемі університету. На основі цього спроєктовано архітектуру рішення, що включає iOS-клієнт для користувацької частини, серверну частину, веб-клієнт для адміністративної частини та модель зовнішнього сервісу ЄДЕБО.

Серверна частина реалізує роботу з профілем вступника, заявами, рейтинговими даними, push-сповіщеннями, підписками та запитами на підтвердження наміру вступу. iOS-клієнт інтегровано в мобільний застосунок SmartUKMA, а веб-клієнт надає працівникам приймальної комісії інструменти для роботи з рейтинговими списками, повідомленнями та відповідями вступників.

Розроблену систему протестовано за допомогою unit-тестів, інтеграційних тестів і ручного проходження основних сценаріїв. Отримане рішення демонструє можливість інтеграції кабінету вступника в SmartUKMA та створює основу для подальшого розвитку цифрової взаємодії університету зі вступниками.

ВСТУП

Вступна кампанія є одним із ключових процесів у роботі закладу вищої освіти, оскільки саме в цей період університет уперше взаємодіє з майбутніми студентами. Вступник обирає освітні програми, подає заяви, визначає їх пріоритетність, стежить за статусами, аналізує своє місце у рейтингових списках, очікує рекомендації до зарахування та повинен вчасно підтвердити вибір місця навчання. Кожен із цих етапів має чіткі строки й безпосередньо впливає на результат вступу.

В Україні основним джерелом даних вступної кампанії є Єдина державна електронна база з питань освіти. Саме в ЄДЕБО зберігається інформація про вступників, конкурсні пропозиції, заяви, статуси, рейтингові списки та результати вступу. Для вступу на бакалаврат важливу роль відіграють результати національного мультипредметного тесту, а конкурсний бал залежить від спеціальності, оскільки для різних предметів застосовуються різні вагові коефіцієнти. Водночас для вступника взаємодія з цими даними часто залишається недостатньо зручною: необхідно самостійно перевіряти оновлення, слідкувати за дедлайнами, відстежувати зміну статусів і повідомлення від приймальної комісії.

Актуальність теми полягає в необхідності створення зручного мобільного інструменту для вступника, який дозволяє швидко отримувати актуальну інформацію про власні заяви, рейтинги, зміни статусів, рекомендації до зарахування та строки підтвердження вибору місця навчання. Для Національного університету «Києво-Могилянська академія» логічною платформою для такого рішення є SmartUKMA — інформаційна система університету, що вже об’єднує окремі цифрові сервіси та підтримує розвиток єдиного користувацького середовища. Інтеграція кабінету вступника в SmartUKMA дозволяє розширити наявну цифрову екосистему університету новим функціональним модулем для взаємодії зі вступниками.

Метою роботи є розроблення кабінету вступника в мобільному застосунку SmartUKMA для iOS із серверною підтримкою, веб-клієнтом для адміністративної частини та інтеграцією з моделлю зовнішнього сервісу ЄДЕБО.

Об’єктом дослідження є процес цифрової взаємодії вступника з університетом під час вступної кампанії.

Предметом дослідження є методи та засоби розроблення мобільного кабінету вступника, серверної логіки обробки вступних даних, push-сповіщень, рейтингових списків та веб-клієнта для адміністративної частини.

Для досягнення поставленої мети необхідно розв’язати такі завдання: проаналізувати процес вступу до закладу вищої освіти та роль ЄДЕБО як основного джерела вступних даних; визначити проблеми, що виникають під час взаємодії вступника з університетом у межах вступної кампанії; дослідити місце кабінету вступника в екосистемі SmartUKMA; спроєктувати архітектуру рішення, що включає iOS-клієнт, серверний сервіс, mock EDEBO та веб-клієнт для адміністративної частини; визначити API-контракти між мобільним клієнтом, веб-клієнтом для адміністративної частини, серверною частиною та моделлю зовнішнього сервісу ЄДЕБО; реалізувати серверну частину для роботи із заявами, рейтинговими списками, push-сповіщеннями, підписками та запитами на підтвердження вступу; реалізувати iOS-клієнт для користувацької частини в застосунку SmartUKMA; реалізувати веб-клієнт для адміністративної частини, якою користуються працівники приймальної комісії; провести тестування розробленої системи на рівні серверної частини, iOS-клієнта для користувацької частини та веб-клієнта для адміністративної частини.

У процесі виконання роботи використовувалися методи аналізу предметної області, моделювання програмної архітектури, проєктування API-контрактів, розроблення програмних компонентів та тестування. Для уточнення взаємодії із зовнішнім джерелом вступних даних було

проаналізовано доступні матеріали, пов’язані з ЄДЕБО, наукові роботи та відкриті репозиторії, у яких описано або використано відповідні API-контракти.

Практичне значення роботи полягає у створенні прототипу, який пов’язує мобільний кабінет вступника, серверну обробку вступних даних та веб-клієнт для адміністративної частини в єдиний робочий сценарій. Запропоноване рішення зменшує залежність від ручної комунікації зі вступниками, дає приймальній комісії інструменти для персональних і масових повідомлень, а також дозволяє фіксувати відповіді вступників на запити про підтвердження наміру вступу.

Робота складається з анотації, вступу, трьох розділів, висновків, списку використаних джерел та додатків. У першому розділі розглянуто вступну кампанію в контексті цифрового університету. У другому розділі описано проєктування кабінету вступника: вимоги до системи з урахуванням ролей користувачів, архітектуру рішення, API-контракти та клієнтські інтерфейси. У третьому розділі наведено імплементацію серверної частини, iOS-клієнта для користувацької частини, веб-клієнта для адміністративної частини, інтеграцію зі SmartUKMA та результати тестування.

РОЗДІЛ 1. ВСТУПНА КАМПАНІЯ В КОНТЕКСТІ ЦИФРОВОГО УНІВЕРСИТЕТУ

1.1 Процес вступу до закладу вищої освіти

Вступ до закладу вищої освіти є послідовним процесом, у межах якого вступник проходить кілька взаємопов’язаних етапів: підготовку до вступних випробувань, створення електронного кабінету, вибір конкурсних пропозицій, подання заяв, участь у конкурсному відборі, отримання рекомендації та підтвердження вибору місця навчання. Цей процес є формалізованим і регулюється Порядком прийому на навчання для здобуття вищої освіти, який визначає правила вступної кампанії, строки її проведення, умови конкурсного відбору та процедуру зарахування.

У вступній кампанії беруть участь декілька основних сторін. Вступник формує власну вступну стратегію: обирає спеціальності та освітні програми, оцінює результати вступних випробувань, подає заяви й визначає їх пріоритетність. Заклад вищої освіти, зокрема приймальна комісія, організовує прийом, опрацьовує заяви, забезпечує комунікацію зі вступниками та здійснює зарахування після виконання встановлених вимог. Центральним джерелом даних у цьому процесі є Єдина державна електронна база з питань освіти, у якій відображаються конкурсні пропозиції, заяви, статуси, рейтингові списки, рекомендації та результати зарахування.

Однією з базових сутностей вступної кампанії є конкурсна пропозиція. Вона описує можливість вступу на певну спеціальність або освітню програму в конкретному закладі освіти, а також пов’язана з формою навчання, джерелом фінансування, кількістю місць та іншими умовами вступу. Для вступника конкурсна пропозиція є практичною одиницею вибору: саме на неї подається заява, саме в її межах розраховується конкурсний бал і формується рейтинговий список.

Заява вступника пов’язує конкретного вступника з обраною конкурсною пропозицією. Вона містить дані про спеціальність, заклад освіти, форму участі в конкурсі, пріоритет, статус опрацювання та інші параметри, що впливають на подальший перебіг вступу. У межах вступної кампанії вступник може подати декілька заяв, тому для нього важливо бачити не лише сам факт подання, а й поточний стан кожної заяви окремо. У 2026 році для вступу на бакалаврат передбачено подання до десяти заяв загалом, з яких не більше п’яти можуть бути подані на місця державного замовлення.

Важливою характеристикою заяви є її пріоритетність. Пріоритет показує, яка з поданих заяв є для вступника більш бажаною. Найвищим є пріоритет 1, і саме він відображає найбільш бажану для вступника освітню траєкторію. Після того як заява отримує статус зареєстрованої у закладі освіти, змінити її пріоритет уже неможливо. Через це пріоритетність не є допоміжним технічним полем: вона безпосередньо впливає на стратегію вступника і має бути зрозуміло відображена в інформаційних системах, якими він користується.

Статус заяви є основним індикатором її поточного стану. Зміна статусу показує, чи заяву подано, зареєстровано, допущено до конкурсу, рекомендовано до зарахування, відхилено, скасовано або включено до наказу. Для вступника статус заяви є не просто інформаційним повідомленням, а сигналом до подальших дій. Наприклад, рекомендація до зарахування потребує своєчасного підтвердження вибору місця навчання, а відхилення або скасування заяви може змінити його фактичні шанси на вступ за іншими заявами.

Для вступу на бакалаврат основним критерієм конкурсного відбору є результати національного мультипредметного тесту. НМТ у 2026 році складається з чотирьох предметних блоків: української мови, математики, історії України та одного предмета на вибір вступника. Результати цих предметів переводяться у шкалу 100–200 балів і використовуються для

розрахунку конкурсного бала. Таким чином, НМТ виступає не ізольованим іспитом, а джерелом вхідних даних для подальшого конкурсного відбору.

Конкурсний бал розраховується з урахуванням предметних коефіцієнтів, які залежать від спеціальності. Це означає, що однакові результати НМТ можуть давати різний підсумковий бал для різних спеціальностей. Наприклад, для технічних спеціальностей більшу вагу може мати математика або фізика, тоді як для гуманітарних напрямів — українська мова, історія України або іноземна мова. Через це конкурсний бал не можна розглядати як універсальну характеристику вступника: він завжди має сенс у контексті конкретної конкурсної пропозиції.

На основі конкурсних балів, пріоритетів, статусів заяв та правил прийому формуються рейтингові списки. Рейтинговий список показує місце вступника серед інших осіб, які подали заяви на ту саму конкурсну пропозицію. Для вступника це один із найважливіших індикаторів вступної ситуації, оскільки він дозволяє оцінити власну позицію відносно інших учасників конкурсу. Водночас рейтинг є динамічним: він може змінюватися внаслідок оновлення даних, появи нових заяв, зміни статусів, відмов або підтверджень інших вступників.

Електронний кабінет вступника на порталі ЄДЕБО є основним офіційним інструментом, через який вступник працює зі вступними даними. Через нього можна подавати заяви, реєструватися на окремі вступні випробування у випадках, передбачених Порядком прийому, відстежувати статуси заяв і підтверджувати вибір місця навчання. Отже, ЄДЕБО виконує роль центральної системи, у якій зберігається й оновлюється офіційна інформація про вступну кампанію, а електронний кабінет є основним каналом доступу вступника до цієї інформації.

Після оприлюднення списків рекомендованих до зарахування вступник повинен виконати вимоги для зарахування у встановлений строк. Одним із ключових кроків є підтвердження вибору місця навчання. Після цього може

укладатися договір про навчання, зокрема із застосуванням електронного підпису, а остаточне зарахування здійснюється наказом керівника закладу освіти. Для вступника завершенням цього ланцюжка є зміна статусу заяви на такий, що підтверджує включення до наказу.

Таким чином, процес вступу до закладу вищої освіти складається не з однієї дії, а з послідовності станів і рішень: складання НМТ, створення електронного кабінету, вибір конкурсних пропозицій, подання заяв, визначення пріоритетів, участь у рейтингу, отримання рекомендації, підтвердження вибору місця навчання та зарахування. Кожен із цих етапів має власні дані, строки та наслідки для вступника. Саме тому інформаційна система, що працює з вступними даними, має не лише відображати окремі поля, а й допомагати вступнику орієнтуватися в логіці всього процесу.

1.2 Кабінет вступника у цифровому університеті

Після розгляду загальної логіки вступної кампанії важливо окремо виділити роль вступника як активного учасника цього процесу. Вступник не лише подає заяви й очікує результату, а постійно приймає рішення на основі даних, що змінюються: обирає конкурсні пропозиції, визначає пріоритети, відстежує статуси заяв, порівнює власне місце у рейтингових списках, реагує на рекомендації та має вчасно підтвердити вибір місця навчання. Помилка або затримка на будь-якому з цих етапів може вплинути на результат вступу.

Особливістю вступного процесу є те, що вступник взаємодіє з університетом ще до того, як стає студентом і отримує повноцінний доступ до внутрішніх університетських сервісів. У цей період він уже потребує персоналізованої інформації, але формально ще не є частиною студентської спільноти. Через це між офіційною вступною інфраструктурою та цифровим середовищем університету виникає проміжна зона, у якій вступник має

отримувати актуальні дані й повідомлення, але ще не користується більшістю сервісів, доступних студентам.

Офіційним інструментом роботи зі вступними даними залишається електронний кабінет вступника в ЄДЕБО. Саме там подаються заяви, відображаються їх статуси та виконується офіційне підтвердження вибору місця навчання. Проте електронний кабінет ЄДЕБО є загальнодержавною системою, а не каналом комунікації конкретного університету зі своїми вступниками. Він виконує функцію офіційної фіксації вступних даних, але не закриває всі потреби університету в оперативному, персоналізованому та мобільному інформуванні вступника.

Для вступника проблема полягає не тільки в наявності або відсутності даних, а й у зручності доступу до них. Дані про заяви, рейтинги, статуси та рекомендації мають практичне значення лише тоді, коли вступник бачить їх вчасно і розуміє, яку дію потрібно виконати далі. Якщо інформація розпорошена між різними каналами або потребує постійної ручної перевірки, зростає ризик пропустити зміну статусу, дедлайн підтвердження або повідомлення приймальної комісії. Саме тому кабінет вступника має розглядатися не просто як екран з даними, а як інструмент супроводу вступника під час всієї вступної кампанії.

У контексті НаУКМА природним середовищем для такого інструменту є SmartUKMA. SmartUKMA є корпоративною інформаційною платформою університету, яка розвивається як набір взаємопов’язаних сервісів для підтримки навчальних, адміністративних та комунікаційних процесів. Її архітектура побудована за мікросервісним принципом: окремі бізнес-функції реалізуються у вигляді самостійних сервісів, що взаємодіють між собою через API. Такий підхід дозволяє додавати нові функціональні модулі без повної перебудови вже наявної системи.

У наявній екосистемі SmartUKMA вже присутні сервіси, пов’язані з інформацією про користувачів, освітні програми, вступні етапи, новини,

документи та повідомлення. Веб-клієнт SmartUKMA використовується як адміністративне середовище для роботи з університетськими даними, а мобільний застосунок забезпечує зручний доступ користувачів до частини сервісів з телефону. Тому розроблення кабінету вступника в межах SmartUKMA є логічним продовженням розвитку платформи, а не створенням окремого ізольованого продукту.

Кабінет вступника у SmartUKMA виконує роль університетського шару над вступними даними. Він не замінює ЄДЕБО і не перебирає на себе функції офіційної державної системи. Його призначення полягає в тому, щоб надати вступнику зручний мобільний доступ до власних заяв, рейтингової інформації, статусів і повідомлень приймальної комісії, а університету — канал оперативної взаємодії зі вступником. Такий підхід дозволяє поєднати офіційність даних ЄДЕБО з персоналізованим користувацьким досвідом у цифровій екосистемі університету.

З боку вступника кабінет має підтримувати базові сценарії, пов’язані з переглядом власного профілю, результатів НМТ, поданих заяв, поточних статусів і місця у рейтингових списках. Окреме значення має підписка на оновлення вступної кампанії, оскільки вона дозволяє отримувати повідомлення про важливі зміни без постійного ручного оновлення інформації. Також кабінет має підтримувати сценарій отримання персонального запиту від приймальної комісії, наприклад щодо підтвердження наміру вступу.

Окремою функцією такого кабінету є подання вступнику не лише первинних даних, а й похідної аналітичної інформації. Результати НМТ самі по собі не завжди дають повне розуміння позиції вступника, оскільки конкурсний бал залежить від спеціальності та вагових коефіцієнтів предметів. Тому кабінет вступника може відображати розрахований конкурсний бал у межах конкретної освітньої програми, місце вступника у рейтинговому списку та орієнтовну індикацію ймовірності проходження.

Така індикація не замінює офіційного рішення приймальної комісії або даних ЄДЕБО, але допомагає вступнику краще оцінити власну ситуацію під час вступної кампанії.

З боку приймальної комісії кабінет вступника потребує веб-клієнта для адміністративної частини. Працівники приймальної комісії мають бачити рейтингові списки, знаходити вступників, надсилати персональні повідомлення, виконувати масові розсилки та контролювати відповіді на запити про підтвердження наміру вступу. Це зменшує залежність від неструктурованої комунікації через телефонні дзвінки або окремі повідомлення, а також створює єдину точку роботи з групами вступників у межах вступної кампанії.

Важливою функцією кабінету вступника є підтримка сценарію підтвердження наміру вступу. Університет може звернутися до вступника з персональним запитом і отримати відповідь у мобільному застосунку. При цьому така відповідь не повинна підміняти офіційне підтвердження вибору місця навчання в ЄДЕБО. Вона виконує попередню комунікаційну функцію для приймальної комісії: допомагає зрозуміти намір вступника, організувати подальшу роботу з рекомендованими особами та своєчасно спрямувати вступника до офіційної дії в ЄДЕБО.

Отже, кабінет вступника у цифровому університеті є не просто додатковою сторінкою з інформацією, а інструментом підтримки взаємодії між вступником, університетом і офіційною вступною інфраструктурою. У випадку SmartUKMA цей інструмент природно інтегрується в уже наявну мікросервісну екосистему, розширюючи її на етап, що передує навчанню. Саме така інтеграція дозволяє розглядати вступника як повноцінного користувача цифрового університету ще до моменту зарахування.

Висновки до розділу 1

У першому розділі було розглянуто вступну кампанію як послідовний процес, у якому вступник проходить через подання заяв, визначення пріоритетів, участь у конкурсному відборі, відстеження рейтингових списків, отримання рекомендації та підтвердження вибору місця навчання. Показано, що цей процес спирається на офіційні дані ЄДЕБО, а для вступу на бакалаврат важливу роль відіграють результати НМТ, предметні коефіцієнти та конкурсний бал у межах конкретної конкурсної пропозиції.

Окрему увагу приділено вступнику як активному учаснику вступної кампанії. Для нього важливими є не лише самі дані про заяви, а й своєчасність їх отримання, зрозумілість поточного статусу, доступ до рейтингової інформації та можливість оперативно реагувати на повідомлення приймальної комісії. Через це кабінет вступника має виконувати не тільки інформаційну, а й комунікаційну функцію.

Також обґрунтовано доцільність інтеграції кабінету вступника в екосистему SmartUKMA. Такий підхід дозволяє розглядати вступника як користувача цифрового університету ще до моменту зарахування, а сам кабінет — як університетський шар взаємодії зі вступними даними, що не замінює ЄДЕБО, але доповнює офіційну вступну інфраструктуру зручними мобільними та адміністративними сценаріями. Отримані висновки є основою для подальшого визначення вимог до системи, її API-контрактів та клієнтських інтерфейсів.

РОЗДІЛ 2. ПРОЄКТУВАННЯ КАБІНЕТУ ВСТУПНИКА

2.1. Вимоги до системи

Вимоги до кабінету вступника визначаються з урахуванням двох основних користувацьких ролей: вступника та працівника приймальної комісії. Вступник взаємодіє із системою через iOS-клієнт, інтегрований у мобільний застосунок SmartUKMA, і розглядає кабінет як персональний простір у межах вступної кампанії. Працівник приймальної комісії взаємодіє із системою через веб-клієнт для адміністративної частини та працює не з власними даними, а з множиною вступників, заяв і станів вступного процесу.

Для вступника система повинна забезпечувати доступ до персональних вступних даних у зрозумілій формі. До таких даних належать профіль вступника, результати НМТ, подані заяви, пріоритети, статуси заяв, рейтингові позиції та розрахований конкурсний бал у межах конкретної освітньої програми. Оскільки конкурсний бал залежить від предметних коефіцієнтів спеціальності, система має відображати його не як універсальну характеристику вступника, а як показник, прив’язаний до конкретної програми.

Окремою вимогою для iOS-клієнта є підтримка сценаріїв, пов’язаних зі змінами у вступній кампанії. Вступник повинен мати можливість отримувати push-сповіщення, підписуватися на оновлення вступної кампанії, бачити актуальні статуси заяв та реагувати на персональні повідомлення приймальної комісії. Такий підхід зменшує потребу в постійному ручному оновленні інформації та допомагає вступнику вчасно реагувати на критичні зміни статусів, рекомендації до зарахування та строки підтвердження вибору місця навчання.

Для сценарію підтвердження наміру вступу система повинна підтримувати надсилання вступнику персонального запиту від приймальної комісії, відображення цього запиту в мобільному застосунку, таймер

очікування відповіді та можливість обрати одну з доступних дій. Відповідь вступника в межах SmartUKMA не є офіційним підтвердженням вибору місця навчання в ЄДЕБО, тому після позитивної відповіді система має спрямовувати вступника до офіційного електронного кабінету ЄДЕБО для виконання необхідних дій відповідно до правил вступної кампанії.

Для працівника приймальної комісії система повинна забезпечувати роботу зі вступниками як із керованими групами та списками. Веб-клієнт для адміністративної частини має дозволяти переглядати рейтингові списки, бачити заяви вступників, використовувати пошук і фільтри за програмою або статусом, а також переходити від конкретного вступника до дії: надіслати персональне повідомлення, створити запит на підтвердження наміру вступу або переглянути історію відповідей. Це дозволяє приймальній комісії працювати з вступниками системно, а не через розрізнені ручні канали комунікації.

Система також повинна підтримувати масову комунікацію. Для цього необхідні загальні розсилки для всіх вступників, які підписані на відповідний канал, а також тематичні повідомлення щодо етапів вступної кампанії. Водночас масові запити на підтвердження наміру вступу мають надсилатися не всім вступникам, а лише тим, хто відповідає обраним умовам, наприклад має заяву на певну програму з відповідним статусом.

З боку серверної частини основними вимогами є централізована обробка вступних даних, збереження стану заяв, розрахунок конкурсних балів, формування рейтингових списків, обробка push-токенів, керування підписками та збереження відповідей на запити приймальної комісії. Серверна частина має забезпечувати узгоджений доступ до даних для iOS-клієнта і веб-клієнта для адміністративної частини, щоб бізнес-логіка не дублювалася в окремих клієнтських застосунках.

Оскільки офіційні вступні дані належать до сфери ЄДЕБО, система повинна враховувати зовнішній характер цього джерела. Це означає, що

кабінет вступника не має підміняти офіційний електронний кабінет вступника, а повинен працювати як університетський шар відображення, обробки та комунікації навколо вступних даних. Для проєктування системи важливо передбачити можливість отримання даних про вступників, заяви, спеціальності, результати НМТ, рейтингові списки та статуси з окремого зовнішнього джерела.

Нефункціональні вимоги до системи пов’язані з надійністю, зрозумілістю інтерфейсів та інтеграцією в наявну екосистему SmartUKMA. Мобільний клієнт має залишатися зручним для вступника і не перевантажувати його технічними деталями, веб-клієнт для адміністративної частини має бути придатним для щоденної роботи приймальної комісії, а серверна частина має забезпечувати стабільну взаємодію між клієнтами та моделлю зовнішнього джерела вступних даних.

2.2. Загальна архітектура рішення

Архітектура кабінету вступника побудована як розподілена клієнт-серверна система, інтегрована в екосистему SmartUKMA. Основними компонентами рішення є iOS-клієнт, веб-клієнт, серверна частина, база даних серверної частини та модуль взаємодії із зовнішнім джерелом вступних даних. Такий поділ дозволяє відокремити користувацькі інтерфейси від бізнес-логіки та зберігати правила обробки вступних даних у центральному серверному шарі.

iOS-клієнт виконує роль тонкого клієнта для вступника. Він відповідає за відображення персональних даних, навігацію між основними сценаріями, отримання push-сповіщень і передавання дій користувача на сервер. На рівні мобільного застосунку може використовуватися локальне сховище для кешування окремих даних і збереження локальних станів інтерфейсу, однак воно не є основним джерелом вступних даних і не повинно містити бізнес-логіку, критичну для всієї системи.

Веб-клієнт призначений для працівників приймальної комісії. Він забезпечує адміністративний доступ до даних, необхідних для перегляду рейтингових списків, роботи зі вступниками, надсилання повідомлень і створення запитів на підтвердження наміру вступу. На відміну від iOS-клієнта, веб-клієнт орієнтований не на персональний перегляд одного користувача, а на роботу з множиною записів і керування комунікаційними сценаріями.

Серверна частина є центральним компонентом архітектури. Вона приймає запити від клієнтських застосунків, взаємодіє із зовнішнім джерелом вступних даних, виконує обробку даних і надає клієнтам узгоджену внутрішню модель. Саме на серверному рівні зосереджуються правила розрахунку конкурсних балів, формування рейтингових списків, перевірки умов для розсилок і обробки відповідей вступників.

База даних серверної частини використовується для збереження внутрішнього стану кабінету вступника. До такого стану належать дані, сформовані на основі зовнішнього джерела, push-токени мобільних пристроїв, підписки на повідомлення, запити на підтвердження наміру вступу, відповіді вступників та службові стани системи. Наявність власного сховища потрібна тому, що частина даних виникає вже у взаємодії між вступником і університетом та не є безпосередньою частиною офіційної вступної інфраструктури.

Модуль взаємодії із зовнішнім джерелом вступних даних ізолює клієнтські застосунки від особливостей отримання первинної інформації. Завдяки цьому iOS-клієнт і веб-клієнт працюють не з форматом зовнішнього сервісу напряму, а з даними, підготовленими серверною частиною. Такий підхід спрощує подальшу зміну джерела даних або способу інтеграції без суттєвої зміни клієнтських інтерфейсів.

Комунікаційний шар системи пов’язаний із push-сповіщеннями та підписками. Серверна частина зберігає токени пристроїв, визначає аудиторії

повідомлень і формує події, які мають бути доставлені вступникам. Мобільний клієнт, у свою чергу, відповідає за отримання сповіщення та відкриття відповідного сценарію в застосунку.

Узагальнено архітектуру можна описати як систему, у якій серверна частина є посередником між зовнішнім джерелом даних, серверною базою даних, iOS-клієнтом і веб-клієнтом. Така модель дозволяє використовувати єдину бізнес-логіку для різних клієнтів і створює передумови для подальшого розширення системи, зокрема підключення Android-застосунку або інших клієнтських інтерфейсів.

2.3. API-контракти та модель даних

Проєктування API-контрактів у кабінеті вступника базується на поділі системи на зовнішнє джерело вступних даних, серверну частину та клієнтські застосунки. Зовнішнє джерело надає первинні дані вступної кампанії, серверна частина приводить їх до внутрішньої моделі та доповнює власними станами, а iOS-клієнт і веб-клієнт працюють уже з узгодженими контрактами серверного API. Такий підхід дозволяє не прив’язувати клієнтські інтерфейси до конкретного формату зовнішнього сервісу.

Для уточнення структури зовнішніх даних було проаналізовано доступні матеріали, пов’язані з ЄДЕБО: офіційні інструкції щодо доступу до тестової ЄДЕБО, відкриті репозиторії з реалізаціями взаємодії з ЄДЕБО та наукові роботи, присвячені інтеграції університетських інформаційних систем із цією базою. У цих джерелах простежуються кілька груп даних, необхідних для роботи кабінету вступника: дані особи, подані заяви, результати вступних випробувань, конкурсні пропозиції, статуси заяв і рейтингові списки. Саме ці групи даних були взяті за основу внутрішньої моделі системи.

У межах реалізації використано не пряме підключення до реальної ЄДЕБО, а модель зовнішнього сервісу вступних даних. Вона відтворює не всі

технічні особливості реальної ЄДЕБО, а лише той фрагмент предметної області, який необхідний для роботи кабінету вступника. Тому числові ідентифікатори статусів, назви методів і формат запитів у реалізованому сервісі можуть відрізнятися від реальних API-контрактів ЄДЕБО, однак змістова модель зберігає ключові сутності вступної кампанії.

Контракт між моделлю зовнішнього джерела та серверною частиною передбачає отримання профілю вступника за кодом ЄДЕБО, а також отримання списку вступників для формування рейтингових списків. Профіль вступника містить персональний ідентифікатор, ПІБ, результати НМТ за предметами та перелік поданих заяв. Кожна заява містить ідентифікатор і назву програми, статус, пріоритет, ознаку участі в конкурсі на бюджетне місце та, за наявності, рейтингову позицію. Цей контракт виконує роль адаптованого представлення зовнішніх вступних даних для потреб серверної частини.

Після отримання даних серверна частина формує внутрішню модель кабінету вступника. У ній окремо виділено вступника, результати НМТ, заяви, вагові коефіцієнти предметів для освітніх програм, push-токени, підписки на повідомлення та запити на підтвердження наміру вступу. Таке розділення важливе, оскільки частина даних надходить із зовнішнього джерела, а частина виникає вже внаслідок взаємодії вступника з університетом у SmartUKMA.

Центральною сутністю логічної моделі є вступник. Із ним пов’язані персональні вступні дані, результати НМТ, подані заяви та службові стани, необхідні для роботи мобільного кабінету. Такий поділ дозволяє відокремити дані, що описують особу вступника, від даних, які характеризують його участь у конкурсі за окремими освітніми програмами.

Заява є окремою сутністю, оскільки один вступник може подати декілька заяв на різні освітні програми. Для кожної заяви важливо зберігати її зв’язок із програмою, поточний статус, пріоритет, ознаку бюджетної або

контрактної участі та рейтингову позицію. Завдяки цьому система може показувати вступнику стан кожної заяви окремо, а веб-клієнт — формувати рейтингові списки й фільтрувати вступників за програмою або статусом.

Результати НМТ також винесено в окрему частину моделі, оскільки вони використовуються повторно для розрахунку конкурсного бала в межах різних програм. Для цього система зберігає предметні результати вступника та окремо підтримує вагові коефіцієнти предметів для освітніх програм. Така структура відображає особливість вступного процесу: конкурсний бал залежить не лише від результатів вступника, а й від спеціальності, для якої він обчислюється.

Окремий блок внутрішньої моделі пов’язаний із комунікацією. До нього належать push-токени мобільних пристроїв, підписки на повідомлення про етапи вступної кампанії та запити на підтвердження наміру вступу. Ці дані не надходять безпосередньо із зовнішнього джерела вступних даних, а виникають у процесі взаємодії вступника з університетом через SmartUKMA.

Узагальнено логічну модель можна описати так: вступник має набір результатів НМТ, набір заяв і пов’язані з ним комунікаційні стани; заява належить до конкретної освітньої програми; вагові коефіцієнти визначають правила розрахунку конкурсного бала для програми; запити на підтвердження пов’язують вступника, програму та відповідь на звернення приймальної комісії. Така модель є достатньою для реалізації основних сценаріїв кабінету вступника і водночас не дублює повністю офіційну структуру ЄДЕБО.

Контракт між серверною частиною та iOS-клієнтом орієнтований на персональні сценарії вступника. Мобільний клієнт отримує профіль вступника, подані заяви, розраховані конкурсні бали, рейтингові позиції, статуси заяв та стан підписки на повідомлення. Окремі API-операції використовуються для авторизації за кодом ЄДЕБО, оновлення push-токена, зміни підписки на повідомлення про вступну кампанію та надсилання

відповіді на запит приймальної комісії. Важливо, що iOS-клієнт не розраховує конкурсний бал самостійно, а отримує підготовлені дані від серверної частини.

Контракт між серверною частиною та веб-клієнтом призначений для адміністративних сценаріїв. Через нього веб-клієнт отримує рейтингові списки за освітньою програмою, список заяв із розрахованими балами, інформацію про зареєстрованих у застосунку вступників, а також історію запитів на підтвердження наміру вступу. Крім операцій читання, цей контракт містить дії для надсилання персональних повідомлень, масових розсилок і створення запитів на підтвердження для одного вступника або для групи вступників, відібраних за програмою та статусом заяви.

Таким чином, API-контракти системи побудовані навколо єдиної внутрішньої моделі вступної кампанії. Зовнішнє джерело використовується для отримання первинних вступних даних, серверна частина відповідає за їх нормалізацію, розрахунки та збереження власного стану, а клієнтські застосунки отримують уже підготовлені дані відповідно до своїх сценаріїв. Така організація не означає, що реальну ЄДЕБО можна підключити без змін, але створює архітектурну основу для майбутньої заміни моделі зовнішнього сервісу на адаптер до реальних API-контрактів ЄДЕБО.

2.4. Опис клієнтських інтерфейсів

Клієнтські інтерфейси кабінету вступника поділяються на дві частини: iOS-клієнт для вступника та веб-клієнт для адміністративної частини. Такий поділ відповідає різним ролям користувачів у системі. Вступник працює з персональними даними та окремими діями, що стосуються його заяв, тоді як працівник приймальної комісії працює зі списками, фільтрами, групами вступників і комунікаційними сценаріями.

iOS-клієнт є основною користувацькою частиною кабінету вступника. Першим сценарієм взаємодії є вхід за ідентифікатором вступника із системи

ЄДЕБО. Такий спосіб входу відповідає прототипному характеру рішення та дозволяє прив’язати користувача мобільного застосунку до вступних даних, отриманих із зовнішнього джерела. Після успішного входу застосунок зберігає стан авторизації та завантажує дані вступника з серверної частини.

Головний екран iOS-клієнта виконує роль персонального огляду вступної ситуації. На ньому відображаються дані вступника, результати НМТ, підписка на сповіщення про етапи вступної кампанії та основні розділи кабінету. Дані згруповано за практичними сценаріями вступника: перегляд освітніх програм, аналіз поданих заяв, відстеження конкурсних балів, статусів і рейтингових позицій.

Для перегляду вступних можливостей у мобільному клієнті передбачено розділ усіх програм. Він дає змогу бачити розрахований конкурсний бал вступника для освітніх програм, а також орієнтовну індикацію результату на основі порівняння з мінімальними показниками. Окремо виділено розділ поданих заяв, у якому показуються лише ті програми, на які вступник уже подав заяву. Це важливо, оскільки рейтинг і статус мають практичне значення саме в контексті конкретної поданої заяви.

Окремим сценарієм iOS-клієнта є робота з push-сповіщеннями. Застосунок реєструє push-токен пристрою на сервері, дозволяє вступнику керувати підпискою на повідомлення про вступну кампанію та відкриває відповідні сценарії після отримання сповіщення. Завдяки цьому критичні повідомлення не губляться серед загальної інформації, а можуть одразу вести користувача до потрібної дії.

Найбільш чутливим мобільним сценарієм є підтвердження наміру вступу. Якщо приймальна комісія надсилає відповідний запит, вступник бачить банер у кабінеті або відкриває модальне вікно після push-сповіщення. У модальному вікні відображається назва спеціальності, текст запиту, таймер очікування відповіді та доступні дії. Вступник може підтвердити намір, відмовитися або закрити вікно, щоб повернутися до відповіді пізніше до

завершення встановленого строку. Позитивна відповідь у SmartUKMA не підміняє офіційну дію в ЄДЕБО, тому після неї користувач має перейти до електронного кабінету вступника ЄДЕБО.

Веб-клієнт для адміністративної частини призначений для працівників приймальної комісії та має іншу логіку організації інтерфейсу. Його основою є робота зі списками та фільтрами, а не з персональним кабінетом одного користувача. У межах адміністративної частини виділено три основні напрями: рейтингові списки, розсилки вступної кампанії та запити на підтвердження вступу.

Розділ рейтингових списків дає змогу переглядати заяви вступників у межах обраної освітньої програми. Для цього передбачено вибір програми, фільтр за статусом і пошук за ПІБ або кодом ЄДЕБО. У таблиці відображаються рейтингова позиція, вступник, спеціальність, статус заяви, пріоритет, конкурсний бал, ознака наявності користувача в застосунку та стан останнього запиту на підтвердження. З цього самого списку працівник приймальної комісії може перейти до надсилання персонального повідомлення або створення запиту на підтвердження для конкретного вступника.

Розділ розсилок призначений для комунікації зі вступниками. Він підтримує персональне повідомлення конкретному вступнику та загальну розсилку для визначеної аудиторії. Для персонального повідомлення працівник приймальної комісії може знайти вступника за ПІБ, кодом ЄДЕБО або даними заяви, після чого заповнити заголовок і текст повідомлення. Для загальної розсилки обирається аудиторія, зокрема всі вступники або ті, хто підписався на оновлення вступної кампанії в мобільному застосунку.

Розділ підтвердження вступу підтримує роботу із запитами на підтвердження наміру вступу. Інтерфейс дозволяє сформувати сегмент вступників за програмою та статусом заяви, знайти окремого вступника, встановити тривалість очікування відповіді та надіслати запит одному

вступнику або всім у вибраному сегменті. Нижче відображається історія відповідей із зазначенням статусу, часу надсилання, дедлайну, часу відповіді та коментаря вступника. Це дозволяє приймальній комісії не лише надсилати запити, а й контролювати реакцію вступників у часі.

Таким чином, клієнтські інтерфейси системи спроєктовані відповідно до різних сценаріїв користування. iOS-клієнт зосереджений на персональному досвіді вступника, швидкому доступі до важливих даних і своєчасній реакції на повідомлення. Веб-клієнт для адміністративної частини орієнтований на роботу приймальної комісії зі списками, фільтрами, розсилками та відповідями вступників. Обидва клієнти використовують єдину серверну модель, тому не дублюють бізнес-логіку і залишаються узгодженими між собою.

Висновки до розділу 2

У другому розділі було визначено вимоги до кабінету вступника з урахуванням двох основних ролей: вступника та працівника приймальної комісії. Для вступника система має забезпечувати персональний доступ до вступних даних, поданих заяв, конкурсних балів, рейтингових позицій, сповіщень і запитів на підтвердження наміру вступу. Для працівника приймальної комісії важливими є робота зі списками вступників, фільтрами, рейтинговими даними, персональними повідомленнями, масовими розсилками та історією відповідей.

Було спроєктовано загальну архітектуру рішення, у якій серверна частина виступає центральним шаром між iOS-клієнтом, веб-клієнтом, базою даних, SmartUKMA та моделлю зовнішнього джерела вступних даних. Такий підхід дозволяє зберігати бізнес-логіку на серверному рівні, не дублювати її у клієнтських застосунках і створює основу для майбутньої заміни моделі зовнішнього сервісу на адаптер до реальних API-контрактів ЄДЕБО.

Окремо описано API-контракти та логічну модель даних системи. У межах цієї моделі виділено вступника, результати НМТ, заяви, вагові коефіцієнти предметів, push-токени, підписки та запити на підтвердження наміру вступу. Така модель поєднує дані, що надходять із зовнішнього джерела, з внутрішніми станами SmartUKMA, які виникають у процесі комунікації між університетом і вступником.

Також було описано клієнтські інтерфейси системи. iOS-клієнт орієнтований на персональний досвід вступника та підтримує перегляд вступної інформації, сповіщення і сценарій відповіді на запит приймальної комісії. Веб-клієнт для адміністративної частини орієнтований на роботу зі списками, розсилками та запитами на підтвердження. Отримані проєктні рішення є основою для подальшої імплементації серверної частини, iOS-клієнта для користувацької частини та веб-клієнта для адміністративної частини.

РОЗДІЛ 3. ІМПЛЕМЕНТАЦІЯ ТА ТЕСТУВАННЯ СИСТЕМИ

3.1. Розроблення серверної частини

Серверна частина кабінету вступника реалізована мовою Java з використанням фреймворку Spring Boot. Вона містить REST API, сервісний шар, доступ до бази даних і фонові задачі, а також забезпечує взаємодію між iOS-клієнтом, веб-клієнтом, сервісами SmartUKMA, Firebase Cloud Messaging і моделлю зовнішнього сервісу ЄДЕБО. Серверний шар відповідає за отримання вступних даних, їх збереження у внутрішній моделі, доповнення локальними станами SmartUKMA, обробку повідомлень і підтримку сценарію підтвердження наміру вступу.

Основною точкою входу для вступника є авторизація за кодом ЄДЕБО, реалізована через REST-ендпойнт серверної частини. Під час першого входу сервер перевіряє, чи існує вступник у власній базі даних. Якщо запису ще немає, сервер звертається до моделі зовнішнього сервісу вступних даних, отримує профіль вступника, результати НМТ і подані заяви разом із рейтинговими показниками, після чого зберігає ці дані у внутрішній моделі системи. Якщо вступник уже зареєстрований, сервер повертає дані з власної бази та оновлює локальні стани, пов’язані з мобільним застосунком.

Для взаємодії із зовнішнім джерелом вступних даних використано окремий клієнтський компонент на основі RestTemplate. Він ізолює основну бізнес-логіку від конкретного формату зовнішнього API та повертає серверній частині уніфіковану модель вступника. У межах прототипу зовнішнім джерелом є mock EDEBO, який віддає дані через REST API. Такий підхід дозволяє тестувати повний сценарій роботи кабінету вступника без доступу до реальної ЄДЕБО, але зберігає архітектурну можливість подальшої заміни цього компонента на адаптер до реальних API-контрактів.

Mock EDEBO реалізовано як окремий серверний застосунок, що моделює поведінку зовнішнього джерела вступних даних. Він зберігає

профілі вступників, результати НМТ, перелік поданих заяв, статуси заяв, пріоритети, ознаки участі в конкурсі, конкурсні бали та рейтингові позиції. Для поданих заяв саме mock EDEBO формує рейтингові показники, тому серверна частина кабінету вступника не дублює логіку побудови офіційного рейтингу, а споживає готові значення і доповнює їх локальними станами SmartUKMA.

Внутрішнє збереження даних реалізовано через реляційну базу даних PostgreSQL із використанням Spring Data JPA для доступу до сутностей. У базі даних зберігаються не лише дані, отримані із зовнішнього джерела, а й локальний стан кабінету вступника, який виникає вже під час взаємодії користувача із системою. Це дозволяє відокремити офіційні вступні дані від даних, необхідних для роботи мобільного застосунку та веб-клієнта.

Структура бази даних побудована навколо кількох груп сутностей. До першої групи належать вступники та їхні результати НМТ; до другої — заяви вступників із програмою, статусом, пріоритетом, рейтинговою позицією, загальною кількістю вступників у списку та конкурсним балом, отриманими із моделі зовнішнього сервісу. Окремо зберігаються push-токени пристроїв, підписки на розсилки, запити на підтвердження наміру вступу, час їх надсилання, дедлайн, поточний стан і відповідь вступника. Для допоміжного сценарію оцінки вступних можливостей до подання заяви також зберігаються вагові коефіцієнти НМТ для бакалаврських програм. Зміни структури бази оформлено через Flyway-міграції, що дає змогу відтворювано розгортати схему та послідовно розширювати її під час розвитку функціональності.

Рейтингові дані поданих заяв не обчислюються сервером кабінету вступника як власна бізнес-логіка. У межах прототипу ці показники формуються в mock EDEBO: для кожної заяви зовнішня модель повертає конкурсний бал у контексті програми, рейтингову позицію та загальну кількість вступників у відповідному списку. Сервер кабінету зберігає ці

значення та передає їх клієнтам у внутрішній моделі, не підміняючи собою офіційне джерело рейтингових даних.

Для адміністративного рейтингового списку сервер отримує з mock EDEBO перелік заяв із готовими рейтинговими показниками, після чого доповнює записи локальною інформацією: чи зареєстрований вступник у мобільному застосунку, чи має push-токен і який останній стан запиту на підтвердження наміру вступу. Заяви зі статусами відмови або скасування відображаються окремо в логіці впорядкування, оскільки такі вступники фактично вже не беруть участі в конкурсі. Таким чином, веб-клієнт отримує рейтинг як робочий інструмент приймальної комісії, але джерелом конкурсних показників залишається модель зовнішнього сервісу.

Окремий блок серверної логіки пов’язаний із push-сповіщеннями. Мобільний застосунок після авторизації передає на сервер FCM-токен пристрою, який зберігається в профілі вступника. На основі цього токена сервер може надсилати персональні повідомлення конкретному вступнику. Для масових розсилок сервер використовує вибрану аудиторію: всіх зареєстрованих вступників із push-токеном або лише тих, хто увімкнув підписку на повідомлення про етапи вступної кампанії. Для доставки повідомлень використовується Firebase Cloud Messaging, що забезпечує подальшу передачу push-сповіщень на iOS-пристрої через APNs.

Сценарій запитів на підтвердження наміру вступу реалізовано як окремий життєвий цикл. Працівник приймальної комісії може створити запит для одного вступника або виконати масове створення запитів для вступників із певною програмою та статусом заяви. Для кожного запиту фіксуються програма, час надсилання, тривалість очікування, дедлайн, поточний статус і відповідь вступника. Якщо для тієї самої заяви вже існує активний запит, сервер не створює дубль, а оновлює наявний запит і перезапускає строк очікування.

Для контролю строків відповіді використовується автоматична перевірка прострочених запитів, реалізована через механізм scheduled tasks у Spring. Сервер періодично знаходить запити, строк відповіді для яких завершився, і переводить їх у стан відмови з відповідним службовим повідомленням. Такий механізм важливий для коректної роботи мобільного клієнта та веб-клієнта: вступник не бачить застарілий активний запит, а працівник приймальної комісії отримує в історії зрозумілий результат завершення строку.

Для веб-клієнта сервер надає адміністративні ендпойнти, що дозволяють отримувати рейтингові списки, список заяв із рейтинговими даними, доступні програми та статуси, надсилати повідомлення й створювати запити на підтвердження. Для iOS-клієнта передбачено ендпойнти авторизації за кодом ЄДЕБО, отримання профілю, перегляду вступних результатів, оновлення push-токена, зміни підписки, отримання активного запиту та надсилання відповіді. Такий поділ API відображає різницю між персональними сценаріями вступника та адміністративними сценаріями приймальної комісії.

Отже, серверна частина є центральним шаром реалізованого кабінету вступника. Вона узгоджує дані між клієнтами, SmartUKMA та моделлю зовнішнього сервісу ЄДЕБО, зберігає локальний стан застосунку, керує підписками, доставкою push-сповіщень і запитами на підтвердження наміру вступу. При цьому рейтингові показники поданих заяв залишаються відповідальністю зовнішньої моделі, а iOS-клієнт і веб-клієнт працюють як тонкі клієнти над узгодженим серверним API.

3.2. Розроблення iOS-клієнта для користувацької частини

iOS-клієнт користувацької частини реалізовано в межах наявного мобільного застосунку SmartUKMA. Це дозволило не створювати окремий застосунок для вступника, а інтегрувати кабінет вступника в уже наявну навігаційну структуру університетського мобільного сервісу. З технічної

точки зору клієнтську частину реалізовано мовою Swift із використанням SwiftUI для побудови інтерфейсу та Combine для обробки асинхронних змін стану.

Основним екраном користувацької частини є кабінет вступника, який відкривається як окрема вкладка в застосунку SmartUKMA. Якщо вступник ще не авторизований, замість основного вмісту відображається екран входу за кодом ЄДЕБО. У полі введення застосовано обмеження на числовий формат і довжину коду, що зменшує кількість помилок ще до надсилання запиту на сервер. Після успішної авторизації код ЄДЕБО зберігається локально, а застосунок завантажує профіль вступника, подані заяви, результати НМТ, вступні результати та активний запит на підтвердження, якщо такий існує.

Логіка екрана зосереджена у view model, яка виконує роль посередника між SwiftUI-представленням, локальним сховищем і серверним API. Такий підхід відповідає моделі MVVM, характерній для SwiftUI-застосунків: представлення відповідає за відображення стану, а view model — за завантаження даних, обробку дій користувача та оновлення опублікованих властивостей. Завдяки використанню @Published-властивостей інтерфейс автоматично реагує на зміну даних профілю, заяв, результатів вступу, підписки або активного запиту.

Для взаємодії із серверною частиною реалізовано окремий API-клієнт кабінету вступника. Він містить методи авторизації за кодом ЄДЕБО, отримання профілю, отримання вступних результатів, реєстрації FCM-токена, зміни підписки на повідомлення про етапи вступної кампанії, отримання активного запиту на підтвердження та надсилання відповіді вступника. HTTP-запити виконуються з використанням Alamofire, а результати повертаються у вигляді Combine Future, що дозволяє однаково обробляти успішні відповіді й помилки.

Отримані з сервера дані декодуються у спеціалізовані DTO-моделі: профіль вступника, заява, результат вступного аналізу та запит на підтвердження. Для заяви клієнт отримує статус, пріоритет, конкурсний бал, рейтингову позицію та загальну кількість вступників у списку. Важливо, що iOS-клієнт не формує рейтингові дані самостійно, а лише відображає значення, отримані через серверну частину. Це відповідає загальній архітектурі рішення, у якій мобільний застосунок залишається тонким клієнтом.

У кабінеті вступника реалізовано два основні режими перегляду: список усіх бакалаврських програм і список поданих заяв. У першому режимі вступник бачить програми, для яких серверна частина надала орієнтовні результати вступного аналізу на основі його результатів НМТ. Для зручності передбачено пошук за назвою програми, спеціальністю або факультетом. У другому режимі відображаються лише заяви, які вступник уже подав, разом із конкурсним балом і рейтинговою позицією в межах відповідної програми.

Окремою частиною iOS-клієнта є робота з push-сповіщеннями. Після запуску застосунок налаштовує Firebase, запитує дозвіл на отримання сповіщень, реєструється в APNs і отримує FCM-токен. Якщо вступник авторизований у кабінеті, токен передається на серверну частину та зберігається в його профілі. Це дозволяє надсилати як персональні повідомлення конкретному вступнику, так і повідомлення для груп вступників відповідно до підписок.

У профілі вступника передбачено перемикач підписки на повідомлення про етапи вступної кампанії. Зміна цього перемикача не є лише локальною дією в інтерфейсі: мобільний клієнт надсилає відповідний запит на сервер, а перед цим за потреби повторно реєструє FCM-токен. Завдяки цьому серверна частина може відрізняти загальні персональні повідомлення від тематичних розсилок про перебіг вступної кампанії.

Найбільш важливим інтерактивним сценарієм iOS-клієнта є підтвердження наміру вступу. Якщо сервер повертає активний запит, у кабінеті відображається банер, з якого вступник може відкрити модальне вікно підтвердження. Та саме модальне вікно може бути відкрито не лише після натискання на push-сповіщення, а й безпосередньо з кабінету, якщо вступник вирішив повернутися до запиту пізніше.

Модальне вікно підтвердження містить назву програми, текст звернення, таймер до завершення строку відповіді, необов’язкове поле коментаря та три дії: підтвердити намір, відмовитися або закрити вікно без відповіді. Таймер розраховується на основі дедлайну, отриманого із серверної частини. Якщо час вичерпується, клієнт оновлює активний запит і закриває модальне вікно, а остаточна зміна стану виконується на сервері. Це важливо, оскільки джерелом істини щодо статусу запиту залишається серверна частина.

Після позитивної відповіді вступника мобільний клієнт надсилає на сервер статус підтвердження та, за наявності, текстовий коментар. Після успішної відповіді активний запит прибирається з локального стану, а вступник спрямовується до офіційного електронного кабінету ЄДЕБО для виконання формального підтвердження вибору місця навчання. Таким чином, відповідь у SmartUKMA використовується як попередній комунікаційний сигнал для приймальної комісії, але не підміняє офіційну дію в ЄДЕБО.

Для обробки push-сповіщень про підтвердження вступу використовується AppDelegate. Під час отримання сповіщення застосунок перевіряє тип повідомлення, фіксує локальний прапорець очікуваного відкриття модального вікна та надсилає подію через NotificationCenter. Кореневий екран застосунку реагує на цю подію переходом до вкладки кабінету вступника, після чого view model завантажує активний запит і відкриває відповідний сценарій. Такий механізм дозволяє коректно

обробляти ситуації, коли користувач відкриває застосунок із push-сповіщення або отримує його під час активної сесії.

Отже, iOS-клієнт для користувацької частини виконує функцію персонального інтерфейсу вступника до даних і дій вступної кампанії. Він забезпечує авторизацію за кодом ЄДЕБО, перегляд профілю, результатів НМТ, вступних можливостей і поданих заяв, керування підпискою на повідомлення та відповідь на запити приймальної комісії. При цьому основна бізнес-логіка зберігається на серверному рівні, а мобільний клієнт відповідає за зручне відображення інформації, локальний стан інтерфейсу та передачу дій вступника до серверної частини.

3.3. Розроблення веб-клієнта для адміністративної частини

Веб-клієнт для адміністративної частини реалізовано в межах наявного адміністративного інтерфейсу SmartUKMA. Для цього було додано окремий розділ навігації «Вступна кампанія», який об’єднує сторінки розсилок, підтвердження вступу та рейтингових списків. Такий підхід дозволяє працівникам приймальної комісії працювати з кабінетом вступника в уже знайомому адміністративному середовищі, не переходячи до окремої зовнішньої системи.

Клієнтську частину реалізовано на Angular із використанням standalone-компонентів, Angular Router, FormsModule та спільного сервісу для взаємодії з backend API. Кожна сторінка адміністративної частини є окремим компонентом, але всі вони використовують спільний ApplicantCabinetService. У цьому сервісі зосереджено HTTP-запити для отримання вступників, заяв, рейтингових списків, надсилання push-повідомлень, створення запитів на підтвердження та отримання історії відповідей.

Першою функціональною частиною веб-клієнта є сторінка рейтингових списків. Вона відображає заяви вступників у межах обраної освітньої програми та подає список одразу як рейтинг. Першою колонкою є номер у

рейтингу, далі відображаються ПІБ вступника, код ЄДЕБО, назва спеціальності, статус заяви, пріоритет, конкурсний бал, ознака реєстрації в застосунку та стан останнього запиту на підтвердження. Для зручності роботи передбачено вибір програми, фільтр за статусом і пошук за ПІБ або кодом ЄДЕБО.

На сторінці рейтингових списків особливу увагу приділено тому, щоб працівник приймальної комісії міг швидко перейти від перегляду даних до дії. Для вступників, які зареєстровані в мобільному застосунку, доступні кнопки переходу до персонального повідомлення або створення запиту на підтвердження вступу. Під час такого переходу веб-клієнт передає через параметри маршруту код ЄДЕБО, а для підтвердження також ідентифікатор програми. Завдяки цьому наступна сторінка відкривається вже з попередньо заповненим контекстом.

Другою частиною адміністративного інтерфейсу є сторінка розсилок вступної кампанії. Вона підтримує два сценарії: персональне повідомлення конкретному вступнику та масову розсилку. Для персонального повідомлення працівник приймальної комісії може знайти вступника за ПІБ, кодом ЄДЕБО, програмою або статусом заяви, після чого ввести заголовок і текст повідомлення. Для масової розсилки обирається аудиторія: загальна розсилка для вступників або повідомлення про оновлення вступної кампанії для тих, хто увімкнув відповідну підписку в мобільному застосунку.

Третім сценарієм є робота із запитами на підтвердження наміру вступу. Веб-клієнт дозволяє сформувати сегмент вступників за програмою та статусом заяви, знайти окремого вступника у вибраному сегменті, встановити тривалість очікування відповіді в хвилинах, годинах або днях і надіслати запит одному вступнику або всім вступникам у вибраному статусі. Такий підхід не зводить підтвердження до загальної розсилки, а пов’язує його з конкретною заявою, програмою та станом вступного процесу.

На сторінці підтвердження вступу також відображається історія відповідей. Для кожного запиту показано вступника, програму, поточний статус, час надсилання, дедлайн, час відповіді та коментар вступника. Окремо обробляється автоматична відмова, яка виникає, якщо вступник не відповів до завершення строку. Завдяки цьому працівник приймальної комісії бачить не лише факт відправлення запиту, а й подальшу реакцію вступника в часі.

У веб-клієнті реалізовано візуальну індикацію статусів заяв і запитів. Статуси на кшталт допущення до конкурсу, рекомендації, включення до наказу, відмови або скасування відображаються різними кольорами та бейджами. Для заяв також показується ознака бюджету. Така індикація важлива для адміністративного сценарію, оскільки працівник приймальної комісії працює з великими списками і має швидко розрізняти записи, які потребують уваги.

Важливою особливістю реалізації є синхронізація контексту між сторінками. Рейтингові списки, персональні повідомлення та підтвердження вступу не є ізольованими екранами: працівник може почати з рейтингу, знайти вступника, перейти до повідомлення або запиту на підтвердження, а система збереже потрібні параметри переходу. Це робить адміністративну частину не набором окремих форм, а єдиним робочим сценарієм для приймальної комісії.

Отже, веб-клієнт для адміністративної частини забезпечує працівникам приймальної комісії інструменти для роботи з вступниками, заявами, рейтинговими списками, розсилками та підтвердженнями наміру вступу. Він не виконує самостійної бізнес-логіки формування рейтингу чи станів заяв, а працює з даними, отриманими через серверний API. Основна роль веб-клієнта полягає у зручному представленні адміністративних даних, фільтрації списків і запуску дій, які надалі обробляються серверною частиною.

3.4. Інтеграція з системою SmartUKMA

Інтеграція з SmartUKMA полягала у вбудовуванні розробленого кабінету вступника в уже наявні клієнтські та адміністративні частини системи. Замість створення окремого продукту функціональність було під'єднано до існуючої структури застосунків: користувацький сценарій — до мобільного застосунку, а адміністративний — до веб-клієнта SmartUKMA.

На рівні мобільного застосунку інтеграція виконана через додавання окремого модуля кабінету вступника до наявної навігації. Модуль використовує загальні підходи SmartUKMA до побудови екранів, роботи зі станом і взаємодії з сервером, але має власний контекст користувача, пов'язаний з кодом ЄДЕБО. Завдяки цьому вступник може працювати з кабінетом у межах того самого застосунку, не змішуючи цей сценарій зі студентськими функціями.

Окремою інтеграційною точкою стали push-сповіщення. Кабінет вступника використовує вже наявний у SmartUKMA механізм реєстрації пристрою, отримання FCM-токена та обробки вхідних повідомлень. Для нового функціоналу було додано лише специфічну обробку повідомлень, пов'язаних зі вступною кампанією: персональних сповіщень, тематичних розсилок і запитів на підтвердження наміру вступу.

У веб-клієнті SmartUKMA інтеграція виконана через додавання розділу «Вступна кампанія» до адміністративної навігації. У цьому розділі об'єднано сторінки, які потрібні працівникам приймальної комісії для щоденної роботи: рейтингові списки, персональні й масові повідомлення, а також запити на підтвердження вступу. Таке розміщення дозволяє не відокремлювати новий функціонал від решти адміністративних інструментів університету.

На серверному рівні інтеграція зводиться до узгодження запитів від різних клієнтів. iOS-клієнт і веб-клієнт звертаються до одного backend API, але використовують його для різних сценаріїв: вступник переглядає власні дані та відповідає на запити, а працівник приймальної комісії переглядає

списки, надсилає повідомлення і створює запити. Це дозволяє зберігати спільну бізнес-логіку на серверному рівні, не дублюючи її в клієнтських застосунках.

У підсумку інтеграція з SmartUKMA забезпечила включення кабінету вступника в наявний цифровий контур університету. Розроблена функціональність не існує як ізольований прототип, а пов’язана з мобільним застосунком, адміністративним веб-клієнтом, серверною частиною та моделлю зовнішнього джерела вступних даних через узгоджені сценарії взаємодії.

3.5. Тестування та оцінка результатів

Тестування системи виконувалося на кількох рівнях, оскільки кабінет вступника складається з серверної частини, моделі зовнішнього сервісу ЄДЕБО, iOS-клієнта та веб-клієнта. Основна мета тестування полягала не лише в перевірці окремих методів, а й у підтвердженні того, що ключові сценарії вступника та працівника приймальної комісії працюють узгоджено між різними частинами системи.

Для серверної частини було реалізовано unit-тести сервісного шару та інтеграційні тести REST-контролерів. Unit-тести перевіряють авторизацію вступника за кодом ЄДЕБО, формування профілю, передачу рейтингових показників із зовнішньої моделі, обчислення орієнтовних вступних результатів для програм, логіку підписок і життєвий цикл запитів на підтвердження наміру вступу. Окремо перевірено створення запиту для одного вступника, масове створення запитів за програмою і статусом, повторне надсилання без створення дубліката, позитивну відповідь, відмову, автоматичну відмову після завершення строку та ігнорування відповіді для вже завершеного запиту.

Інтеграційні тести backend API перевіряють поведінку контролерів з погляду клієнтських застосунків. Зокрема, тестуються ендпойнти входу вступника, отримання профілю, створення запиту на підтвердження, масове

надсилання запитів, отримання активного запиту, надсилання відповіді, отримання адміністративних опцій та списку заяв. У таких тестах важливо перевірити не внутрішню реалізацію сервісів, а коректність HTTP-статусів, валідації вхідних даних і структури відповідей.

Оскільки mock EDEBO у цій роботі моделює зовнішнє джерело вступних даних, для нього також було додано окремі тести. Вони перевіряють формування конкурсного бала, призначення рейтингових позицій і перенесення заяв зі статусами відмови або скасування в кінець рейтингового списку. Це важливо для збереження правильної межі відповідальності: рейтингові показники поданих заяв формуються на стороні моделі зовнішнього сервісу, а сервер кабінету вступника лише отримує та передає їх клієнтам.

Веб-клієнт для адміністративної частини протестовано за допомогою unit-тестів Angular-компонентів і спільного сервісу взаємодії з backend API. Тести сторінки рейтингових списків перевіряють побудову списку програм і статусів, фільтрацію за програмою, статусом і пошуковим запитом, індикацію статусів заяв, індикацію стану запиту на підтвердження та переходи до персонального повідомлення або створення запиту. Тести сторінки підтвердження вступу перевіряють завантаження опцій і заяв, обчислення строку очікування в хвилинах, годинах і днях, надсилання запиту одному вступнику, масове надсилання запитів, обробку автоматичної відмови та візуальне відображення статусів. Для сторінки розсилок перевірено персональне повідомлення, масову розсилку, роботу з вибором вступника та реакцію інтерфейсу на успішні й помилкові відповіді сервера.

iOS-клієнт покрито тестами з використанням фреймворку Swift Testing. Автоматизовані тести зосереджені на тих частинах мобільного клієнта, які мають бути стабільними незалежно від стану мережі: декодування відповідей сервера, робота з optional-полями, мапінг імовірності вступу, передача конкурсного бала та рейтингових показників у модель відображення, парсинг

дат у форматах ISO 8601 і Spring LocalDateTime, обчислення часу до завершення запиту та формування тексту банера підтвердження. Це дозволяє зменшити ризик помилок на межі між backend API та мобільним інтерфейсом.

Сценарії, які залежать від реального пристрою, APNs, Firebase Cloud Messaging або взаємодії користувача з модальним вікном, додатково перевірялися вручну. До таких сценаріїв належать отримання дозволу на push-сповіщення, передача FCM-токена на сервер, надсилання повідомлення з адміністративної частини, відкриття модального вікна підтвердження з push-сповіщення та з кабінету вступника, відповідь вступника, відображення помилки у разі невдалого надсилання, автоматичне завершення запиту після дедлайну та повторне надсилання запиту з оновленням строку очікування.

Оцінка результатів виконувалася через проходження основних користувацьких сценаріїв. Для вступника перевірялися вхід за кодом ЄДЕБО, перегляд профілю, результатів НМТ, усіх бакалаврських програм, поданих заяв, рейтингової позиції та активного запиту на підтвердження. Для працівника приймальної комісії перевірялися перегляд рейтингового списку, фільтрація заяв, пошук вступника, надсилання персонального повідомлення, масова розсилка, створення запиту на підтвердження для одного вступника та для групи вступників, а також перегляд історії відповідей.

За результатами тестування було підтверджено, що реалізована система підтримує основні сценарії кабінету вступника: отримання вступних даних із моделі зовнішнього сервісу, відображення їх у мобільному застосунку, адміністративну роботу зі списками вступників, доставку повідомлень, керування підписками та обробку запитів на підтвердження наміру вступу. Виявлені під час перевірки проблеми, зокрема некоректне трактування часу дедлайну, дублювання запитів, зайва технічна інформація в адміністративному інтерфейсі та помилки в мобільному модальному вікні, були виправлені в процесі розроблення.

Висновки до розділу 3

У третьому розділі описано практичну реалізацію кабінету вступника та перевірку його працездатності. Було реалізовано серверну частину, iOS-клієнт для користувацької частини, веб-клієнт для адміністративної частини, mock EDEBO як модель зовнішнього джерела вступних даних та інтеграцію цих компонентів із системою SmartUKMA.

Серверна частина забезпечила роботу з профілями вступників, заявами, рейтинговими даними, вступними результатами, push-сповіщеннями, підписками та запитами на підтвердження наміру вступу. iOS-клієнт надав вступнику персональний інтерфейс для перегляду даних і взаємодії з приймальною комісією, а веб-клієнт забезпечив адміністративні сценарії роботи з рейтинговими списками, повідомленнями та підтвердженнями.

Тестування показало, що реалізовані компоненти узгоджено працюють між собою та підтримують основні сценарії вступної кампанії. Автоматизовані тести охопили серверну логіку, API-контракти, модель зовнішнього сервісу, клієнтську логіку веб-клієнта та критичні моделі iOS-клієнта. Сценарії, що залежать від реального пристрою та push-інфраструктури, були додатково перевірені вручну.

Отже, у межах розділу було підтверджено, що спроєктоване рішення може працювати як цілісний прототип кабінету вступника в екосистемі цифрового університету.

ВИСНОВКИ

У кваліфікаційній роботі було розроблено кабінет вступника в мобільному застосунку SmartUKMA для iOS із серверною підтримкою та веб-клієнтом для адміністративної частини. Робота охопила аналіз предметної області вступної кампанії, проєктування архітектури, реалізацію програмних компонентів, інтеграцію з моделлю зовнішнього сервісу ЄДЕБО та тестування основних сценаріїв.

У процесі аналізу було встановлено, що вступна кампанія є складним процесом, у якому вступник має постійно відстежувати подані заяви, статуси, рейтингові списки, рекомендації до зарахування та строки виконання вимог. Офіційним джерелом таких даних є ЄДЕБО, однак для вступника важливо не лише мати доступ до даних, а й отримувати їх у зручній, персоналізованій і мобільній формі.

У роботі було визначено основні вимоги до кабінету вступника: авторизація за кодом ЄДЕБО, перегляд профілю, результатів НМТ, поданих заяв і рейтингових позицій, орієнтовний аналіз вступних можливостей, підтримка push-сповіщень, підписка на оновлення вступної кампанії, отримання та опрацювання запитів на підтвердження наміру вступу, а також адміністративна робота з вступниками через веб-клієнт.

Для реалізації цих вимог було спроєктовано архітектуру, що складається з iOS-клієнта, серверної частини, веб-клієнта для адміністративної частини та mock EDEBO. Такий поділ дозволив відокремити клієнтське відображення даних від серверної бізнес-логіки, а модель зовнішнього сервісу — від внутрішньої логіки кабінету вступника. Це також створює основу для подальшого підключення інших клієнтів, зокрема Android-застосунку.

У межах реалізації було створено серверну частину на Spring Boot із використанням PostgreSQL, REST API, сервісного шару та механізмів роботи з push-сповіщеннями. Було реалізовано mock EDEBO, який моделює

отримання вступних даних, рейтингових позицій і конкурсних балів поданих заяв. На стороні iOS-клієнта було реалізовано користувацький кабінет вступника, а у веб-клієнті SmartUKMA — адміністративний розділ «Вступна кампанія».

Розроблена система підтримує два взаємопов’язані типи користувачів. Вступник отримує персональний мобільний інтерфейс для роботи зі своїми вступними даними, сповіщеннями та запитами приймальної комісії. Працівник приймальної комісії отримує веб-інструмент для перегляду рейтингових списків, пошуку вступників, надсилання повідомлень і контролю відповідей на запити підтвердження.

Тестування підтвердило працездатність реалізованих компонентів. Unit-тести та інтеграційні тести перевірили серверну логіку, API-контракти та модель зовнішнього сервісу. Angular-тести перевірили ключову логіку адміністративного веб-клієнта. Swift Testing було використано для перевірки DTO-моделей, парсингу дат, обчислення часу до завершення запиту та формування тексту користувацьких повідомлень в iOS-клієнті. Додатково було виконано ручну перевірку сценаріїв, пов’язаних із реальним телефоном і push-сповіщеннями.

Практичний результат роботи полягає у створенні прототипу, який демонструє можливість інтегрувати кабінет вступника в екосистему цифрового університету. Рішення не замінює офіційний електронний кабінет вступника ЄДЕБО, але доповнює взаємодію вступника з університетом через мобільний доступ до даних, оперативні сповіщення та зручний канал комунікації з приймальною комісією.

Отже, поставлену мету роботи досягнуто: розроблено кабінет вступника у мобільному застосунку SmartUKMA для iOS, реалізовано серверну підтримку, веб-клієнт для адміністративної частини, модель зовнішнього сервісу ЄДЕБО та перевірено працездатність системи на основних сценаріях вступної кампанії.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. МОН опублікувало Порядок прийому до закладів вищої освіти у 2026 році. Міністерство освіти і науки України. 24.03.2026. URL: https://mon.gov.ua/news/mon-opublikuvalo-poriadok-pryiomu-do-zakladiv-vyshchoi-osvity-u-2026-rotsi.

2. Гід вступника 2026: покрокова інструкція до здобуття вищої освіти. Міністерство освіти і науки України. 24.03.2026. URL: https://mon.gov.ua/news/hid-vstupnyka-2026-pokrokova-instruktsiia-do-zdobuttia-vyshchoi-osvity.

3. Вступна кампанія. Єдина державна електронна база з питань освіти. URL: https://vstup.edbo.gov.ua/.

4. ЄДЕБО. Міністерство освіти і науки України. URL: https://mon.gov.ua/ministerstvo-2/edebo.

5. Реєстри та сервіси ЄДЕБО. Міністерство освіти і науки України. URL: https://mon.gov.ua/reestri-ta-servisi-edebo.

6. Основне про НМТ 2026. Український центр оцінювання якості освіти. URL: https://testportal.gov.ua/osnovne-pro-nmt-2026/.

7. Інструкції користувача ЄДЕБО. ДП «Інфоресурс». URL: https://www.inforesurs.gov.ua/edebo/instructions/.

8. Доступ до тестової ЄДЕБО. ДП «Інфоресурс». 2019. URL: https://www.inforesurs.gov.ua/wp-content/uploads/2019/07/dostup_do_testovoi_edebo.pdf.

9. vstup.edbo.gov.ua.report: Create report of students by page data. GitHub. URL: https://github.com/laskevych/vstup.edbo.gov.ua.report.

10. python-edbo-connector: EDBO-connector. GitHub. URL: https://github.com/EldarAliiev/python-edbo-connector.

11. ums-edbo: RESTful web service for managing EDBO SOAP web service calls. GitHub. URL: https://github.com/ifnul/ums-edbo.

12. ums-backend: RESTful web service for managing UMS. GitHub. URL: https://github.com/ifnul/ums-backend.

13. Мосійчук В. С. Розробка API для інтеграції з системою ЄДЕБО для сайта KSU24 = Development of an API for Integration with the EDEBO System for the

KSU24 Website Testing : кваліфікаційна робота на здобуття ступеня бакалавра / Херсонський державний університет ; керівник М. С. Львов. Івано-Франківськ, 2025. 59 с. URL: https://ekhsuir.kspu.edu/items/54bcfaf7-1b99-450d-bd8e-46ce70f62549.

14. Макоєдова В. О. Інформатизація процесів вступної кампанії в закладах вищої освіти. Технічні науки та технології. 2023. вип. 1(31). С. 90–97. URL: https://doi.org/10.25140/2411-5363-2023-1(31)-90-97.

15. SwiftUI. Apple Developer Documentation. URL: https://developer.apple.com/documentation/swiftui.

16. UserNotifications. Apple Developer Documentation. URL: https://developer.apple.com/documentation/usernotifications.

17. Firebase Cloud Messaging. Firebase Documentation. URL: https://firebase.google.com/docs/cloud-messaging.

18. Spring Boot Reference Documentation. Spring. URL: https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/.

19. Angular Documentation. Angular. URL: https://angular.dev/overview.

20. PostgreSQL Documentation. PostgreSQL. URL: https://www.postgresql.org/docs/.

21. JUnit 5 User Guide. JUnit. URL: https://junit.org/junit5/docs/current/user-guide/.

22. Swift Testing. Apple Developer Documentation. URL: https://developer.apple.com/documentation/testing.

ДОДАТКИ

Додаток 1

Загальна архітектура системи

Додаток призначений для розміщення загальної архітектурної схеми кабінету вступника та його взаємодії з компонентами SmartUKMA.

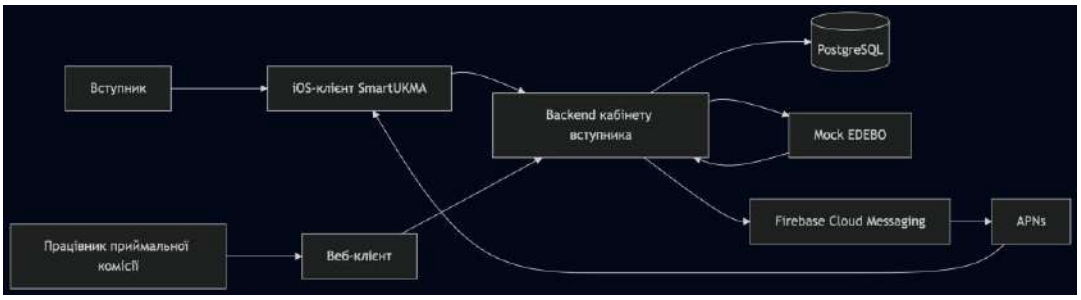

Рисунок 1.1 — Загальна архітектура кабінету вступника в екосистемі SmartUKMA

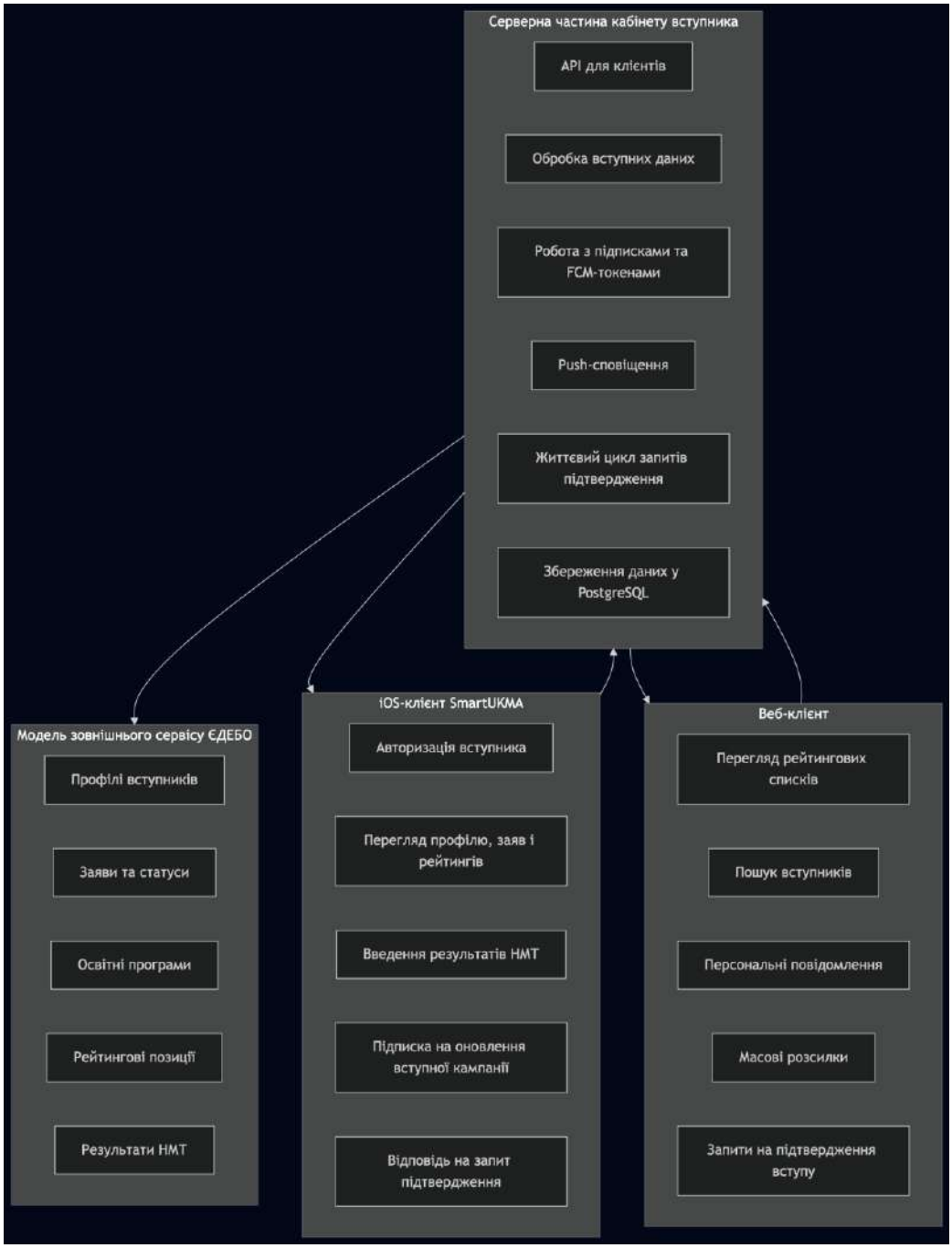

Рисунок 1.2 — Розподіл відповідальності між iOS-клієнтом, веб-клієнтом, серверною частиною та моделлю зовнішнього сервісу ЄДЕБО

Додаток 2

Схема моделі даних

Додаток призначений для схеми основних сутностей, які використовуються для представлення вступника, заяв, освітніх програм, сповіщень і запитів на підтвердження наміру вступу.

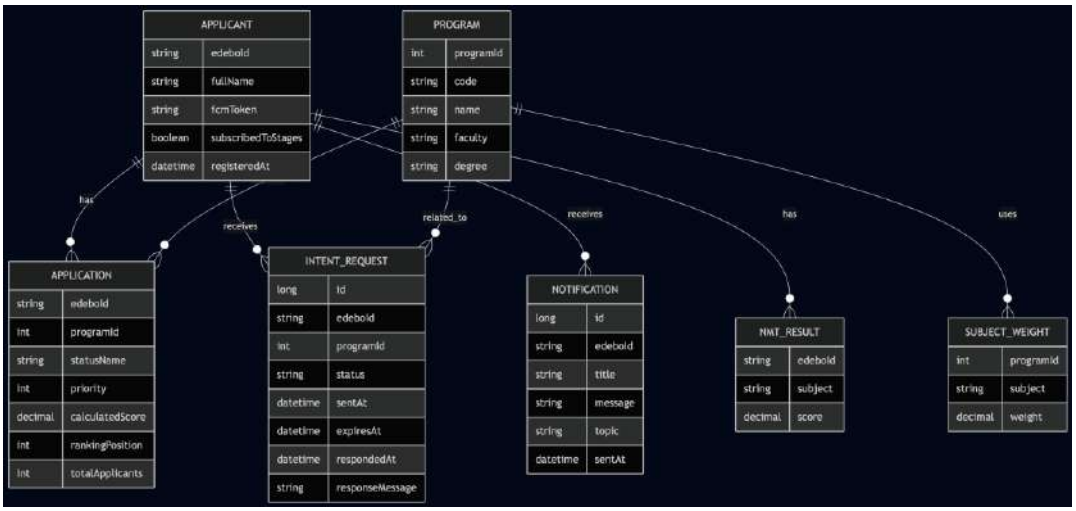

Рисунок 2.1 — Схема основних сутностей кабінету вступника

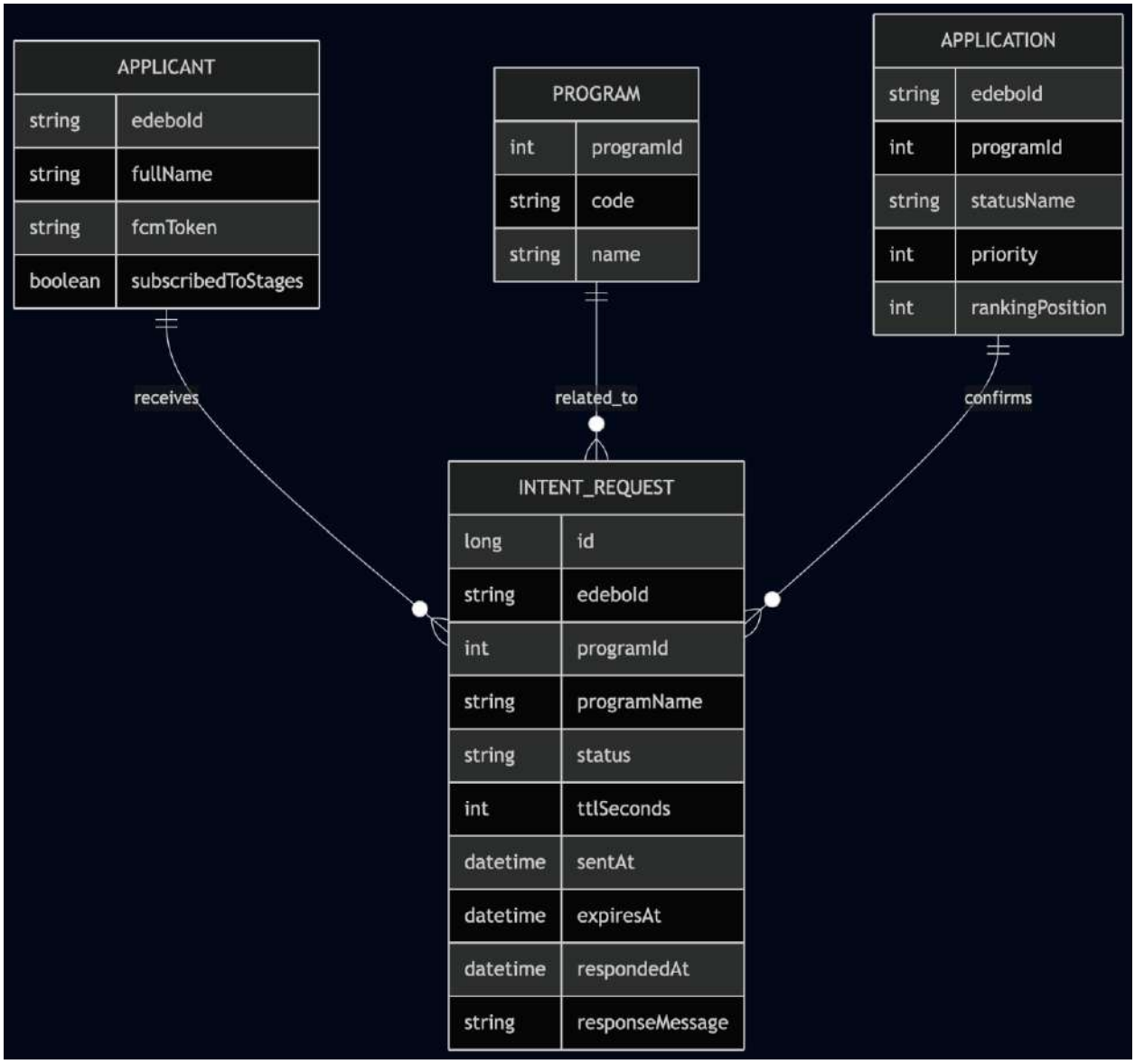

Рисунок 2.2 — Модель даних запиту на підтвердження наміру вступу

Додаток 3

Сценарій підтвердження наміру вступу

Додаток призначений для діаграми послідовності, яка показує створення запиту працівником приймальної комісії, доставку push-сповіщення, відповідь вступника та автоматичну відмову після завершення строку очікування.

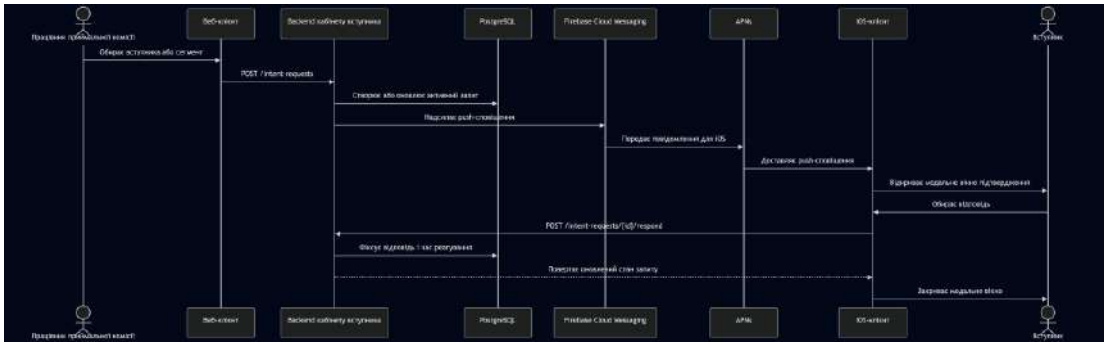

Рисунок 3.1 — Послідовність обробки запиту на підтвердження наміру вступу

Рисунок 3.2 — Стани запиту на підтвердження наміру вступу

Додаток 4

Приклади API-контрактів кабінету вступника

У додатку наведено фрагменти API-контрактів, які використовуються для взаємодії між серверною частиною, iOS-клієнтом і веб-клієнтом.

Лістинг 4.1 — API-контракти для запитів на підтвердження вступу

Файл: applicant-cabinet/src/main/java/.../controller/IntentRequestController.java

```
@PostMapping("/api/v1/intent-requests")
public ResponseEntity<IntentRequestDto> create(
        @Valid @RequestBody CreateIntentRequestDto request) {
    return ResponseEntity.ok(intentRequestService.create(request));
}


@PostMapping("/api/v1/intent-requests/broadcast")
public ResponseEntity<List<IntentRequestDto>> broadcast(
        @Valid @RequestBody BroadcastIntentRequestDto request) {
    return ResponseEntity.ok(intentRequestService.broadcast(request));
}


@GetMapping("/open-api/intent-requests/active/{edeboId}")
public ResponseEntity<IntentRequestDto> getActive(@PathVariable String edeboId) {
    return intentRequestService.getActiveForApplicant(edeboId)
            .map(ResponseEntity::ok)
            .orElseGet(() -> ResponseEntity.noContent().build());
}


@PostMapping("/open-api/intent-requests/{id}/respond")
public ResponseEntity<IntentRequestDto> respond(
        @PathVariable Long id,
        @Valid @RequestBody RespondIntentRequestDto request) {
    return ResponseEntity.ok(intentRequestService.respond(id, request));
}
```

Лістинг 4.2 — API-контракти для персональних і масових сповіщень

Файл: applicant-cabinet/src/main/java/.../controller/NotificationController.java

```
@PostMapping("/send")
public ResponseEntity<BooleanResponse> send(
        @Valid @RequestBody NotificationRequest request) {
    notificationService.sendToApplicant(request);
    return ResponseEntity.ok(new BooleanResponse(true));
}


@PostMapping("/broadcast")
public ResponseEntity<BooleanResponse> broadcast(
        @Valid @RequestBody BroadcastRequest request) {
    notificationService.broadcast(request);
    return ResponseEntity.ok(new BooleanResponse(true));
}
```

Лістинг 4.3 — Типізований контракт веб-клієнта для запиту на підтвердження

Файл: ui-server/web-src/src/app/pages/applicant-cabinet/applicant-cabinet.service.ts

```
export interface CreateIntentRequest {
```

```
  edeboId: string;
  programId: number;
  ttlSeconds: number;
}


export interface BroadcastIntentRequest {
  programId: number;
  statusId: number;
  ttlSeconds: number;
}


export interface IntentRequest {
  id: number;
  edeboId: string;
  applicantFullName?: string;
  programId: number;
  programName?: string;
  status: IntentRequestStatus;
  sentAt?: string;
  expiresAt?: string;
  respondedAt?: string;
  responseMessage?: string;
}
```

Додаток 5

Фрагменти реалізації серверної логіки

У додатку наведено фрагменти серверної логіки, пов’язаної зі створенням, оновленням і завершенням запитів на підтвердження наміру вступу.

Лістинг 5.1 — Оновлення активного запиту замість створення дубліката

Файл: applicant-cabinet/src/main/java/.../service/IntentRequestService.java

```
        private IntentRequestEntity persistOrRefresh(String edeboId, Integer programId,
                                                     String programName, Integer ttlSeconds) {
            LocalDateTime now = LocalDateTime.now();
            Optional<IntentRequestEntity> existing = intentRequestRepository
                    .findFirstByEdeboIdAndProgramIdAndStatusOrderBySentAtDesc(
                            edeboId, programId, IntentRequestStatus.PENDING);


            if (existing.isPresent()) {
                IntentRequestEntity entity = existing.get();
                entity.setProgramName(programName);
                entity.setTtlSeconds(ttlSeconds);
                entity.setSentAt(now);
                entity.setExpiresAt(now.plusSeconds(ttlSeconds));
                entity.setRespondedAt(null);
                entity.setResponseMessage(null);
                return intentRequestRepository.save(entity);
            }


            IntentRequestEntity entity = new IntentRequestEntity();
            entity.setEdeboId(edeboId);
            entity.setProgramId(programId);
            entity.setProgramName(programName);
            entity.setStatus(IntentRequestStatus.PENDING);
            entity.setTtlSeconds(ttlSeconds);
            entity.setSentAt(now);
            entity.setExpiresAt(now.plusSeconds(ttlSeconds));
            return intentRequestRepository.save(entity);
        }
```

Лістинг 5.2 — Обробка відповіді вступника

Файл: applicant-cabinet/src/main/java/.../service/IntentRequestService.java

```
        public IntentRequestDto respond(Long id, RespondIntentRequestDto request) {
            IntentRequestEntity entity = intentRequestRepository.findById(id)
                    .orElseThrow(() -> new IllegalArgumentException("Intent request not found"));


            if (entity.getStatus() != IntentRequestStatus.PENDING) {
                return mapper.toDto(entity);
            }


            if (entity.getExpiresAt() != null && entity.getExpiresAt().isBefore(LocalDateTime.now()))
{
                declineExpired(entity);
                return mapper.toDto(entity);
            }


            entity.setStatus(request.getStatus());
            entity.setRespondedAt(LocalDateTime.now());
            entity.setResponseMessage(request.getResponseMessage());
```

```
        return mapper.toDto(intentRequestRepository.save(entity));
    }
```

Лістинг 5.3 — Автоматична відмова після завершення строку очікування

Файл: applicant-cabinet/src/main/java/.../service/IntentRequestService.java

```
    public void markExpired() {
        LocalDateTime now = LocalDateTime.now();
        intentRequestRepository
                .findByStatusAndExpiresAtBefore(IntentRequestStatus.PENDING, now)
                .forEach(this::declineExpired);
    }


    private void declineExpired(IntentRequestEntity entity) {
        entity.setStatus(IntentRequestStatus.DECLINED);
        entity.setRespondedAt(LocalDateTime.now());
        entity.setResponseMessage(AUTO_DECLINE_MESSAGE);
        intentRequestRepository.save(entity);
    }
```

Додаток 6

Фрагменти реалізації mock EDEBO

У додатку наведено фрагменти логіки моделі зовнішнього сервісу ЄДЕБО, яка використовується для формування рейтингових даних у межах тестового середовища.

Лістинг 6.1 — Збагачення заяв рейтинговими позиціями

Файл: edebo-mock/src/main/java/.../service/RankingEnrichmentService.java

```
public void enrich(Collection<ApplicantProfile> applicants) {
    Map<Integer, List<Application>> byProgram = applicants.stream()
            .flatMap(applicant -> applicant.getApplications().stream())
            .collect(Collectors.groupingBy(Application::getProgramId));


    byProgram.forEach((programId, applications) -> {
        List<Application> active = applications.stream()
                .filter(application -> !isRejected(application))
                .sorted(Comparator.comparing(Application::getCalculatedScore).reversed())
                .toList();


        for (int i = 0; i < active.size(); i++) {
            active.get(i).setRankingPosition(i + 1);
            active.get(i).setTotalApplicants(active.size());
        }
    });
}
```

Лістинг 6.2 — Винесення заяв зі статусами відмови з активного рейтингу

Файл: edebo-mock/src/main/java/.../service/RankingEnrichmentService.java

```
private boolean isRejected(Application application) {
    String status = Optional.ofNullable(application.getStatusName())
            .orElse("")
            .toLowerCase();


    return status.contains("відмова")
            || status.contains("відхил")
            || status.contains("скасовано");
}
```

Додаток 7

Фрагменти реалізації iOS-клієнта

У додатку наведено фрагменти iOS-клієнта, пов’язані з авторизацією вступника, підпискою на сповіщення, отриманням активного запиту та надсиланням відповіді.

Лістинг 7.1 — API-клієнт для активного запиту на підтвердження

Файл: KMASmartApp/Data/HTTP/ApplicantCabinet/ApplicantCabinetApiClient.swift

```
        func getActiveIntentRequest(edeboId: String) -> Future<IntentRequestResponse?, NSError> {
            let url = URL(string: "\(baseUrl)/open-api/intent-requests/active/\(edeboId)")!
            return Future { promise in
                AF.request(url, method: .get)
                    .validate(statusCode: 200..<300)
                    .response { response in
                        if let status = response.response?.statusCode, status == 204 {
                            promise(.success(nil))
                            return
                        }
                        switch response.result {
                        case .success(let data):
                            guard let data = data, !data.isEmpty else { promise(.success(nil)); return }
                            do {
                                let decoded = try JSONDecoder().decode(IntentRequestResponse.self, from: data)
                                promise(.success(decoded))
                            } catch {
                                promise(.failure(NSError(domain: "decode", code: -1)))
                            }
                        case .failure(let error):
                            promise(.failure(NSError(domain: "", code: error.responseCode ?? 0)))
                        }
                    }
            }
        }
```

Лістинг 7.2 — Надсилання відповіді вступника

Файл: KMASmartApp/Data/HTTP/ApplicantCabinet/ApplicantCabinetApiClient.swift

```
        func respondIntentRequest(id: Int, status: String, message: String?)
            -> Future<IntentRequestResponse, NSError> {
            let url = URL(string: "\(baseUrl)/open-api/intent-requests/\(id)/respond")!
            var body: [String: Any] = ["status": status]
            if let message = message, !message.isEmpty {
                body["responseMessage"] = message
            }
            return HttpRequestFactory.shared.performRequest(
                to: url,
                method: .post,
                parameters: body,
                requiresAuth: false
            )
        }
```

Лістинг 7.3 — Оновлення підписки на зміни вступної кампанії

Файл: KMASmartApp/Presentation/Stories/ApplicantCabinet/ApplicantCabinetViewModel.swift

```
        func setSubscription(_ newValue: Bool) {
            guard newValue != isSubscribedToStages else { return }
            Messaging.messaging().token { [weak self] token, error in
                guard let self else { return }
                let registerToken: AnyPublisher<Void, NSError>
                if let token, !token.isEmpty, error == nil {
                    registerToken  =  self.apiClient.registerFcmToken(edeboId:  self.edeboId,  token:
token)
                        .eraseToAnyPublisher()
                } else {
                    registerToken = Just(())
                        .setFailureType(to: NSError.self)
                        .eraseToAnyPublisher()
                }


                registerToken
                    .flatMap  {  self.apiClient.updateSubscription(edeboId:  self.edeboId,  subscribed:
newValue) }
                    .receive(on: DispatchQueue.main)
                    .sink(receiveCompletion: { _ in },
                          receiveValue: { [weak self] _ in self?.isSubscribedToStages = newValue })
                    .store(in: &self.bag)
            }
        }
```

Додаток 8

Фрагменти реалізації веб-клієнта адміністративної частини

У додатку наведено фрагменти веб-клієнта, які відповідають за рейтингові списки, перехід до надсилання повідомлень і створення запитів на підтвердження.

Лістинг 8.1 — Фільтрація рейтингового списку

Файл: ui-server/web-src/src/app/pages/applicant-cabinet/ranking-lists/ranking-lists.component.ts

```
get visibleApplications(): IntentApplication[] {
  const q = this.query.trim().toLowerCase();
  return this.applications.filter(app => {
    if (this.selectedProgramId && app.programId !== this.selectedProgramId) return false;
    if (this.selectedStatusId && app.statusId !== this.selectedStatusId) return false;
    if (!q) return true;
    return [
      app.fullName,
      app.edeboId,
    ].some(value => (value || '').toLowerCase().includes(q));
  });
}
```

Лістинг 8.2 — Перехід із рейтингу до повідомлення або підтвердження

Файл: ui-server/web-src/src/app/pages/applicant-cabinet/ranking-lists/ranking-lists.component.ts

```
writeNotification(app: IntentApplication): void {
  this.router.navigate(['/applicant-cabinet/push-notifications'], {
    queryParams: { edeboId: app.edeboId },
  });
}


requestConfirmation(app: IntentApplication): void {
  this.router.navigate(['/applicant-cabinet/intent-requests'], {
    queryParams: {
      edeboId: app.edeboId,
      programId: app.programId,
    },
  });
}
```

Лістинг 8.3 — Перетворення тривалості очікування у секунди

Файл: ui-server/web-src/src/app/pages/applicant-cabinet/intent-requests/intent-requests.component.ts

```
get ttlSeconds(): number {
  const amount = Math.max(1, Number(this.filters.durationAmount) || 1);
  switch (this.filters.durationUnit) {
    case 'DAYS':
      return Math.round(amount * 24 * 60 * 60);
    case 'HOURS':
      return Math.round(amount * 60 * 60);
    case 'MINUTES':
    default:
      return Math.round(amount * 60);
  }
}
```

Лістинг 8.4 — Створення персонального запиту на підтвердження

Файл: ui-server/web-src/src/app/pages/applicant-cabinet/intent-requests/intent-requests.component.ts

```
sendSelected(): void {
  const app = this.selectedApplication;
  if (!app || !app.programId || !this.filters.durationAmount) return;


  this.sendingSelected = true;
  this.service.createIntentRequest({
    edeboId: app.edeboId,
    programId: app.programId,
    ttlSeconds: this.ttlSeconds,
  }).subscribe({
    next: () => {
      this.toast.success(`Надіслано запит вступнику ${app.edeboId}`);
      this.sendingSelected = false;
      this.reload();
    },
    error: (err) => {
      this.toast.error(err?.error?.message || 'Помилка надсилання');
      this.sendingSelected = false;
    },
  });
}
```

Додаток 9

Скріншоти користувацького та адміністративного інтерфейсів

Додаток призначений для фінальних скріншотів інтерфейсів після завершення візуального полірування. У роботу варто додавати лише екрани, які демонструють фінальний стан системи.

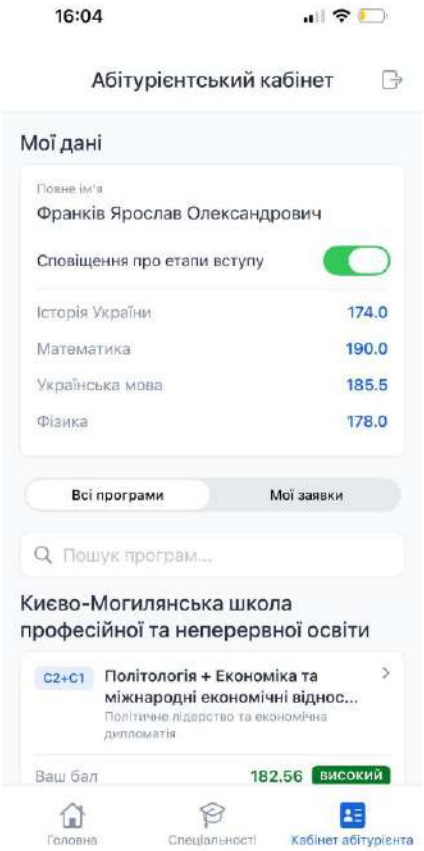

Рисунок 9.1 — Екран кабінету вступника в iOS-клієнті

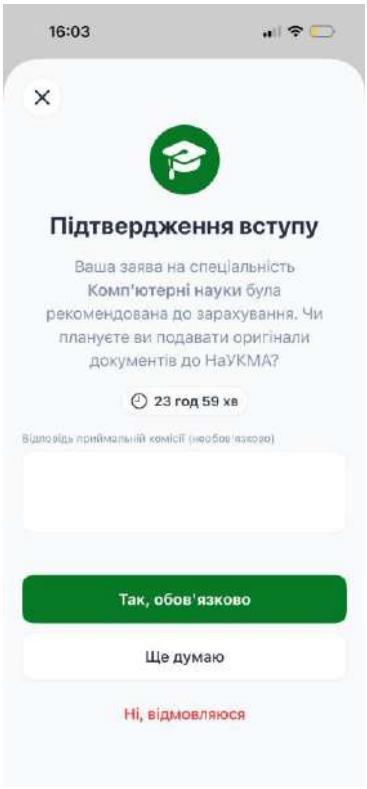

Рисунок 9.2 — Модальне вікно підтвердження наміру вступу в iOS-клієнті

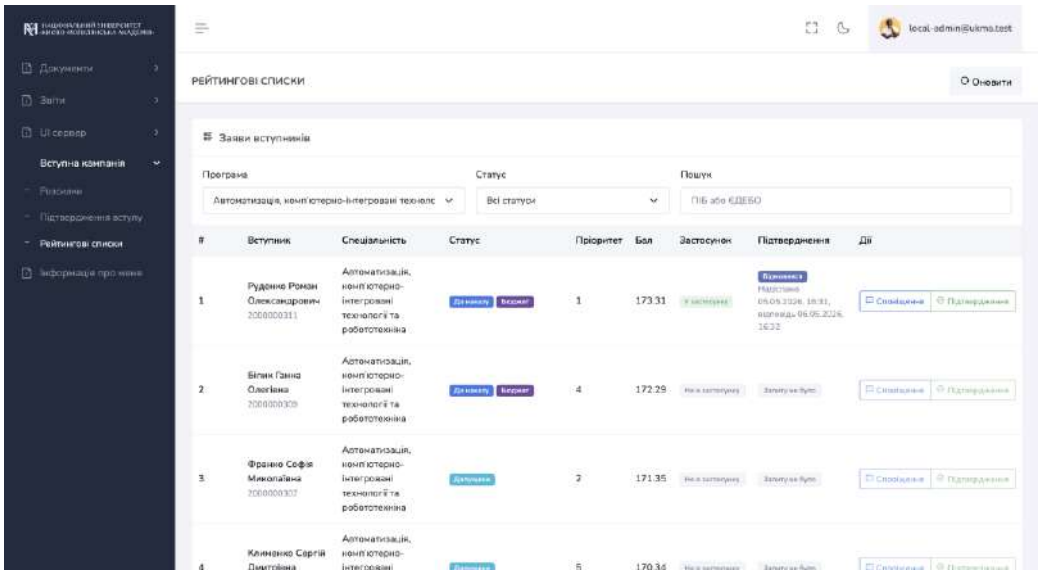

Рисунок 9.3 — Рейтингові списки у веб-клієнті адміністративної частини

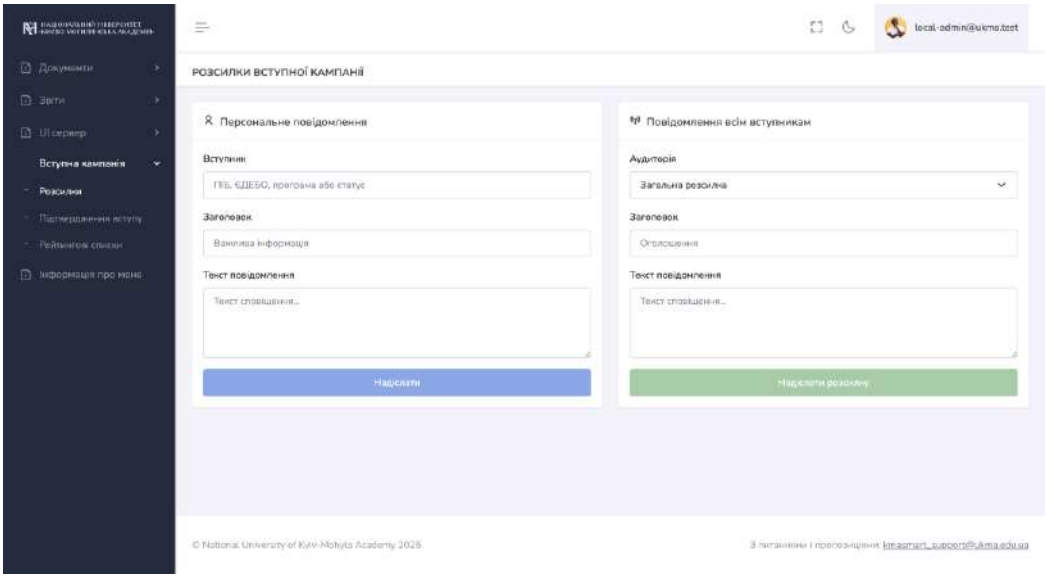

Рисунок 9.4 — Сторінка розсилок вступної кампанії у веб-клієнті адміністративної частини

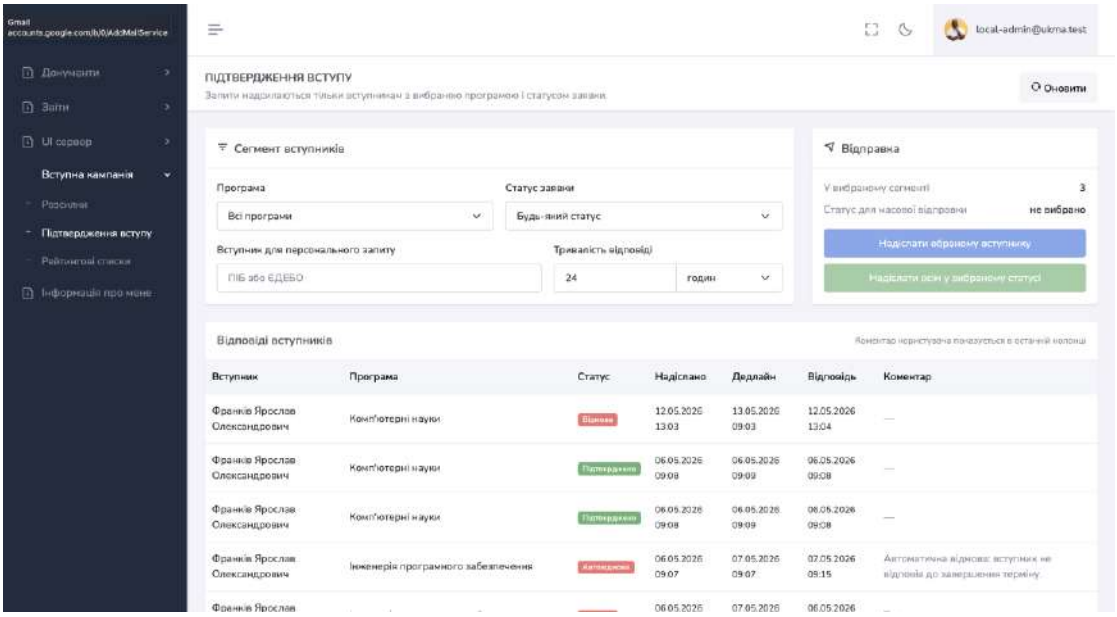

Рисунок 9.5 — Сторінка запитів на підтвердження вступу у веб-клієнті адміністративної частини

Додаток 10

Основні сценарії тестування

Додаток призначений для переліку сценаріїв тестування, які підтверджують коректність взаємодії між backend, iOS-клієнтом, веб-клієнтом і моделлю зовнішнього сервісу ЄДЕБО.

Таблиця 10.1 — Сценарії тестування користувацької частини

Сценарій	Очікуваний результат
Авторизація вступника за кодом ЄДЕБО	Після введення коректного коду відкривається кабінет вступника з персональними даними.
Завантаження профілю вступника	У кабінеті відображаються ПІБ вступника, результати НМТ та пов’язані заяви.
Перегляд поданих заяв	Для кожної заяви відображаються освітня програма, статус, пріоритет і рейтингові дані.
Перегляд рекомендованих освітніх програм	Система показує розрахований конкурсний бал і орієнтовну індикацію ймовірності проходження.
Пошук освітньої програми	Список результатів фільтрується відповідно до введеного тексту.

Таблиця 10.2 — Сценарії тестування сповіщень і підтвердження вступу

Сценарій	Очікуваний результат
Реєстрація FCM-токена після авторизації	Сервер зберігає FCM-токен вступника.
Увімкнення підписки на оновлення вступної кампанії	Значення підписки оновлюється на сервері.
Отримання push-сповіщення про підтвердження	На пристрої відкривається модальне вікно підтвердження.

Позитивна відповідь вступника	Запит отримує статус «Підтверджено», фіксується час відповіді.
Відмова вступника	Запит отримує статус «Відмова», фіксується текст відповіді.
Завершення строку відповіді	Система автоматично фіксує відмову.

Таблиця 10.3 — Сценарії тестування адміністративної частини

Сценарій	Очікуваний результат
Перегляд рейтингового списку за освітньою програмою	Відображається список вступників у рейтинговому порядку.
Фільтрація рейтингу за статусом заяви	У таблиці залишаються лише заяви з обраним статусом.
Пошук вступника за ПІБ або кодом ЄДЕБО	Список фільтрується відповідно до введеного запиту.
Перехід із рейтингу до персонального повідомлення	Форма повідомлення відкривається з попередньо заповненим кодом ЄДЕБО.
Перехід із рейтингу до запиту на підтвердження	Форма запиту відкривається з попередньо заповненим вступником і освітньою програмою.
Масове надсилання запитів за програмою і статусом	Запити створюються лише для вступників, які відповідають обраному сегменту.