Державний вищий навчальний заклад
“Прикарпатський національний університет імені Василя Стефаника”
Кафедра інформаційних технологій

УДК 004

ДИПЛОМНИЙ ПРОЕКТ

Тема Розробка голосового асистента для університету під Android

Спеціальність 121 Інженерія програмного забезпечення

ПОЯСНЮВАЛЬНА ЗАПИСКА

ДП.ПІ-17.ПЗ
(позначення)

Рецензент
проф. Кузь М. В.
(посада) (підпис) (дата) (розшифровка підпису)

Студент
ІПЗ-41 Петелюк Л. Б.
(шифр групи) (підпис) (дата) (розшифровка підпису)

Нормоконтролер
проф. Кузь М. В.
(посада) (підпис) (дата) (розшифровка підпису)

Керівник дипломного проекту
доц. Ткачук В. М.
(посада) (підпис) (дата) (розшифровка підпису)

Допускається до захисту

Завідувач кафедри

Козленко М. І.
(посада) (підпис) (дата) (розшифровка підпису)

2020
(рік)

Державний вищий навчальний заклад
«Прикарпатський національний університет імені Василя Стефаника»
Факультет математики та інформатики Кафедра інформаційних технологій
Спеціальність Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ:
Завідувач кафедри Козленко М. І.
„_____”_____ 20___р.

ЗАВДАННЯ
НА ВИКОНАННЯ ДИПЛОМНОГО ПРОЕКТУ

Студенту Петелюку Любомирові Богдановичу
(прізвище, ім’я, по батькові студента)

1. Тема проекту Розробка голосового асистента для університету під Android затверджена розпорядженням по факультету математики та інформатики від „25” жовтня 2019р.№7
2. Термін здачі студентом закінченого проекту 22 травня 2020 р.
3. Вихідні дані до дипломного проекту клієнтська частина проекту – голосовий асистент у вигляді Android-додатку, серверна частина проекту – API для роботи з розкладом

4. Зміст пояснювальної записки (перелік питань, що їх належить опрацювати)
   1. Аналіз предметної області та постановка задачі
   2. Проектування програми
   3. Реалізація поставленого завдання
   4. Бізнес-план

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

6. Дата видачі завдання _____

Керівник _____ Ткачук В.М.
(підпис) (розшифровка підпису)

Завдання прийняв до виконання _____ Петелюк Л. Б.
(підпис) (розшифровка підпису)

КАЛЕНДАРНИЙ ПЛАН

Номер і назва етапів дипломного проекту	Термін виконання етапів проекту	Примітка
1. Аналіз предметної області та постановка задачі	02.12.2019	
2. Проектування програми	15.02.2020	
3. Реалізація поставленого завдання	17.04.2020	
4. Бізнес-план	11.05.2020	
5. Оформлення пояснювальної записки	18.05.2020	

Студент _____ Петелюк Л.Б.
(підпис) (розшифровка підпису)

Керівник проекту _____ Ткачук В. М.
(підпис) (розшифровка підпису)

РЕФЕРАТ

Пояснювальна записка: 90 сторінок (без додатків), 41 рисунків, 24 джерел.

Ключові слова: метод, функція, рядок, об’єкт, клас, пакет, список, масив, асоціативний масив, випадаючий список, Front-end, Back-end.

Об`єктом дослідження є Android-додаток, голосовий асистент.

Мета роботи: розробити голосовий асистент для роботи з розкладом навчальних пар університету для мобільний пристроїв на базі операційної системи Android.

Стислий опис тексту пояснювальної записки:

Розробка проекту ділиться на розробку Front-end та Back-end частини. Front-end частина – Android-додаток, голосовий асистент. Back-end частина – API для роботи з розкладом навчальних пар університету.

ABSTRACT

Explanatorynote: 90 pages (without appendix), 41 figures, 24 references.

Key words: method, function, string, object, class, package, list, array, associative array, dropdown, Front-end, Back-end

Object of study: Android application, voice assistant.

Brief description of the text of the explanatory note:

Project development is divided into Front-end and Back-end part development. Front-end part - Android-application, voice assistant. The back-end part is an API for working with the schedule of university study pairs.

ЗМІСТ

ВСТУП

На сьогоднішній момент існують дві проблеми в IT сфері пов’язані з Прикарпатським національним університетом імені Василя Стефаника та розробкою голосових асистентів на IT ринку.

Перша полягає у відсутності сучасного програмного продукту для мобільних платформ призначеного для роботи з навчальним розкладом університету який міг би видати результати на глобальніші питання у межах даної теми. Наприклад, студенти чи викладачі не можуть дізнатись список вільних аудиторій в межах конкретного факультету і часу в короткий проміжок часу та в зручній формі. Для цього їм прийдеться використовувати доступні аналоги та тратити доволі велику кількість часу для обробки великої кількості даних.

Друга проблема полягає у відсутності попиту голосових асистентів для комерційних проектів. Сьогодні існує кілька голосових асистентів від топових виробників, таких як Google, Microsoft, Amazon, Яндекс, та інших. Більшість з них націлена на споживача і несе в більшості розважальний характер. Через відсутність комерційних проектів пов’язаних із розробкою голосових асистентів для торгівельних компаній, підприємців, малого та середнього бізнесу голосові асистенти сприймаються людьми як дорога іграшка, або непотрібний функціонал на мобільному пристрої.

Метою проекту є розробка голосового асистенту для роботи з розкладом навчальних пар університету у вигляді Android-додатку. Додаток зможе обробляти велику кількість даних та з допомогою голосових команд отримуватиме інформацію, яку не зможуть дати аналоги проекту у вигляді веб-додатку чи телеграм-боту.

Об'єкт дослідження – голосовий асистент в якості комерційного проекту та полегшення побутового життя студентів та викладачів у межі університету

шляхом розробки програмного забезпечення для роботи з розкладом навчальних пар.

Предметом дослідження є голосовий асистент у вигляді Android-додатку.

Актуальність предмету доволі велика, так як дає можливість отримувати відповіді на більш масштабні запити (наприклад дізнатись список вільних аудиторій), що полегшить побутове життя в межах навчального закладу. Крім цього проект використовує API в якому безпосередньо знаходиться логіка роботи з розкладом – це дозволить розробити проект для різних платформ, навіть для електрочайника.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Аналіз ринку голосових асистентів

Як зазначалось тут [1, 2, 3, 4] віртуальні голосові асистенти доволі широко використовуються на смартфонах і не тільки починаючи з 2011 року. На сьогоднішній момент люди з високою регулярністю відправляють велику кількість запитів різного характеру: “включи музику”, “встанови нагадування”, “надішли повідомлення”, “перевір погоду” “перевір список справ”, та навіть запити нестандартного характеру: “Я тобі подобаюсь?”. Проте, незважаючи на велику популярність та корисність, голосові асистенти програють мобільним додаткам та інтернету на ІТ-ринку. Ситуація змінюється. Гіганти ринку представляють інновації для широкого кола важливих напрямків, що доволі швидко призведе до трансформації на ринку інформаційних технологій.

Суттєве зростання голосових асистентів можна спостерігати з кінця 2014 року, коли компанія Amazon розробила та вивела на світ гаджети Alexa та AmazonEcho. В основному таке зростання завдячує популяризації смарт-гучномовців та відновленням зацікавленості до голосових асистентів на мобільних девайсах. Хоч голосові асистенти і не досконалі на сьогоднішній день, але сама система розпізнавання, зчитування, обробки та генерації голосу є інноваційною, що дуже приваблює споживачів і маркетологів

Згідно із даними досліджень компанії Canalys станом на 2018 рік, ринок смарт-динаміків у світі виріс більше ніж у півтори рази. Тут компанія Google виходить у лідери, так як їх пристрої встановлені в більше 5 мільйонів житлах. Amazon посідає друге місце і має більше 4 мільйонів проданих девайсів Echo. В загальному, компанія Amazon до сих пір лідирує на ринку смарт-гучномовців у Сполучених Штатах Америки з часткою 62 відсотки, але ця частка менша у порівнянні з початком року, коли було більше 70%. З виходом на ринок нових

виробників, наприклад Apple і їх HomePod, частка Echo зменшується. Крім цього, нові моделі Echo користуються попитом, як правило, тільки у власників пристроїв даного бренду, тому кількість користувачів AmazonEcho, скоріш за все, продовжуватиме зменшуватись.

Проте, компанії Amazon з цього приводу поки що не варто хвилюватись, адже ринок доволі швидко зростає та в довгостроковій перспективі частина ринку не буде мати велике значення. Врешті-решт, найбільш важливо зберегти лояльність клієнтів.

Amazon не показує будь-яких ознак зменшення ритму ініціативи AlexaEverywhere, що доволі непогано демонструє бажання комерційного електронного мастодонта стати універсальним маркетом EverythingStore. Кілька місяців тому Vendor представив світові у своїх девайсах режим Show, який трансформує планшети Fire HD на EchoShow для того, щоб працювати у бездротовому режимі з Alexa. Крім цього, компанія Amazon представла телевізійну приставку, що у свою чергу постачається з функціями Echo, має назву: Fire TV Cube. Також виробник почав впроваджувати версію Alexa, видозмінену для готельної справи, бажаючи розробити голосового асистента для спілкування з гостями. Marriott, до речі, є однією з перших мереж готельної сфери, що впровадила Alexa в якості віртуального дворецького.

Дані зусилля Amazon вказують на їх зацікавленість у створенні бази користувачів і встановленні їхнього голосового асистента в якості стандарту для розумного житла. Нові покоління користувачів через Alexa програватимуть музику, отримуватимуть прогнози погоди, робитимуть телефонні дзвінки, і, що важливіше, будуть робити покупки товарів на сайті компанії. Проте, недавні звіти про дуже малий рівень популярності покупок через Alexa, що становить всього лиш 2% вказують на те, що функціонал голосових асистентів використовується менше порівняно з очікуваннями аналітиків даної галузі. Не зважаючи на це, Amazon не поспішає покидати ринок голосових асистентів для

шопінгу, оскільки Alexa разом із пристроями Echo є лише частиною довгострокового плану компанії по захопленню IT-ринку для домогосподарств.

Найбіль вагомою новою функцією, що Google інтегрував до свого голосового асистенту, є функція, націлена на покупки через Google Assistant. Програма “Акції та Покупки” спрямована на розробку єдиної мережі для покупок у межах голосового асистенту та сервісу для покупок Google Express. Дане рішення огортатиме велику кількість технологій, що будуть включати у собі функції пошуку продуктів, мобільного шопінгу та голосового пошуку.

Кінцева мета Google - сформувати групу роздрібних торговців, на відміну від Amazon, пропонуючи потужну торгову систему на платформі для об’єднання рітейлерів.

Таке положення Google добре видно в незвичному оновленні програмного забезпечення для GoogleAssistant. Пошуковий гігант оприлюднив свою програму GoogleDuplex, щоб продемонструвати, як неймовірно точний GoogleAssistant може імітувати людську мову. Google позиціонує Duplex як функцію, яка може допомогти користувачам зателефонувати в організацію, щоб забронювати готель чи ресторан, придбати квитки - і все від імені людини. Ваші підписники навіть не знають, що спілкуються з GoogleAssistant. Неважко уявити, що Інтернет дуже швидко поставив питання про потенційні етичні проблеми, які могли б призвести до таких “розмов з роботом”.

Взагалі, Google, схоже, знає, як зробити чудові проривні продукти, але не має добре розробленої стратегії виведення голосових продуктів на ринок.

Зрештою, найбільша сила Google полягає у її здатності використовувати свої послуги пошуку даних для розширення можливостей продуктів штучного інтелекту. Але незрозуміло, як Google використовуватиме їх для забезпечення найкращого голосового досвіду.

Окрім дуополії Amazon-Google, цього року ринок смарт-динаміків почав штурмувати Apple із гаджетом HomePod. Сьогодні такий девайс становить 6% ринку згідно з даними StrategyAnalytics. Але HomePod - це просто високий клас

динаміків, який постачається із вбудованим асистентом Siri для простих голосових команд. Це не настільки потужне рішення, як AmazonEcho та GoogleHome, тому навряд чи має сенс порівнювати їх.

Однак є деякі ознаки того, що Apple готується зробити Siri більш відкритою для сторонніх розробників. Це дозволить конкурувати з Alexa та GoogleAssistant та зробить HomePod більш корисним пристроєм. Однією з найбільш захоплюючих нових можливостей iOS 12 є функція швидкого доступу, яка дозволяє користувачам Siri встановлювати власні фрази для часто використовуваних програм або навіть налаштовувати цілу послідовність дій для управління кількома програмами, які можна запустити за допомогою однієї голосової команди.

Після налаштування цей ярлик також працюватиме на HomePod та AppleWatch. Порівняно з Alexa та Google Assistant, які вимагають від користувачів вивчити певні фрази для активації голосового пристрою, Apple надає користувачам більше гнучкості та пропонує розробникам спосіб інтегрувати свої додатки в Siri, тим самим покращуючи можливості HomePod. Крім того, деякі останні звіти також дозволяють припустити, що Siri незабаром зможе отримати підтримку багатьох користувачів. Таке явище безумовно підвищить у кілька разів корисність HomePod.

На даний момент Siri відстає від GoogleAssistant та AmazonAlexa за своїми функціональними можливостями, багато в чому завдяки рішенню компанії Apple не надавати до відкритого доступу вихідний код Siri для розробників, здатних створювати голосові програми. Але важливо пам’ятати, що Siri є одним із найпопулярніших голосових асистентів з надійною базою споживачів - щомісяця обробляється 10 мільярдів запитів; крім того, Apple вже закладає основу для використання голосових технологій поза розумними динаміками з підтримкою SiriAppleWatch та AirPods. Жоден її конкурент ще не звернув на це уваги. Тепер, коли компанія вживає серйозних заходів для відкриття Siri та вдосконалення її функціональності за допомогою “ярликів”,

Apple може дуже швидко стати потужним гравцем у конкуренції голосових інтерфейсів.

Згідно з повідомленнями ЗМІ, Facebook розпочав розробку функції розпізнавання голосу для своєї флагманської програми та месенджера під назвою Aloha та подає заявку на патент на доповідача, який зосередиться на відеочаті та комунікаціях. Компанія навіть планувала ввести голосовий інтерфейс для Instagram, щоб полегшити процес обміну миттєвими повідомленнями. Однак поки немає конкретної інформації про те, коли фактично будуть запущені ці функції. Ми, мабуть, почуємо деякі новини з Facebook до кінця цього року.

Незважаючи на те, що репутація Facebook у США погіршилася після недавнього скандалу з приводу витоку особистих даних, на багатьох світових ринках соціальна мережа відчуває себе досить впевнено. Це означає, що Facebook може запустити свої голосові продукти на закордонні ринки та оцінити їх ефективність перед тим, як представити їх на ринок США.

Але сьогодні китайські бренди швидко ростуть на світовому ринку смарт-ораторів. І хоча в першому кварталі 2018 року Amazon та Google складали близько 70% ринку смарт-ораторів, у попередньому кварталі їх частка досягла 94% (за даними StrategyAnalytics). Частково це пояснюється сильним зростанням ринку китайських смарт-ораторів, якого не вистачає як в Amazon, так і в Google. І хоча Siri можна використовувати в Китаї, HomePod немає. Місцеві гіганти Alibaba та Xiaomi є лідерами в цьому сегменті ринку в Китаї, і їх продажів вистачає, щоб увійти до п’ятірки кращих світових виробників. Розумний помічник Alibaba під назвою TmallGenie навіть безпосередньо інтегрований в управління автомобілями Mercedes, Audi та Volvo, які продаються в Китаї.

Минулого тижня Amazon та Microsoft почали інтегрувати системи Alexa та Cortana, що дозволило Cortana та Alexa працювати спільно для обміну між собою даними та надання індивідуальних відповідей, в незалежності від того,

який віртуальний помічник запитує користувач. Така співпраця відкриває цікаві перспективи для голосових помічників та вказує на вектор розвитку екосистеми голосової команди.

За замовчуванням на ноутбуках Windows Cortana отримує доступ до індивідуальних даних користувачів в професійному плані, тобто їх робочі контакти і так далі, в той час як Alexa зазвичай отримує особисті дані користувача, не пов’язані з його професійною діяльністю. Разом таке партнерство мусить забезпечити повноцінну роботу клієнтів у їхньому професійному та особистому житті. Microsoft вважає, що в майбутньому віртуальні помічники працюватимуть разом для користувачів, але в той же час кожен з них матиме власну спеціалізацію. Це означає, що це справді реально, принаймні в короткостроковій перспективі, але намагайтеся, поки ви не побачите єдиного віртуального помічника, який буде задоволений усіма аспектами нашого життя. Години роботи, цифрові індикатори, які мають більшу швидкість, менший пристрій. Хоча сьогодні продавець просуває власне рішення на ринку.

Сьогодні на більшості популярних платформ віртуального помічника користувач може надсилати запити та працювати лише з обмеженим набором вбудованих сервісів. А якщо ви хочете працювати з сторонніми додатками, вам потрібно додати ім'я постачальника послуг до запиту, а потім використовувати його спеціальний набір команд. В результаті запити виглядають приблизно так: “Помічнику, попросіть “Додаток 7” виконати “команду 5” ” . Користувачам важко запам’ятати численні назви служб та дізнатися їх набори команд, така модель погана. Користувачі все ще працюють лише з невеликою кількістю вбудованих сервісів, які з самого початку підтримували помічник.

Надалі віртуальні помічники матимуть більш інтегрований та інтегрований інтерфейс, який дозволяє користувачам задавати будь-які запитання у будь-якій зручній формі та більш природно взаємодіяти з

послугами сторонніх розробників. З точки зору користувача, зручніше взаємодіяти з одним помічником, який здатний виконувати безліч різних завдань, замість великої кількості різних помічників, кожен з яких має свої можливості, управління, пам'ять тощо. Якщо це передбачення здійсниться, користувачі отримають більш ефективний та налаштований інструмент, а постачальники послуг матимуть легко масштабований канал для доступу до своїх послуг..

На даний момент, додаючи нові послуги до платформи голосових помічників, не рідко існує велика різниця між інструментами, доступ до яких мають великі компанії, що спеціалізуються на розробці віртуальних помічників, та ресурсами, доступ до яких мають сторонні розробники. Останнім доступні веб-інструменти, що можуть забезпечити лише основний аналіз природного мови та прості шаблони для побудови діалогу.

Надалі сторонні розробники нарешті отримають доступ до складних платформ та інструментів з більшою функціональністю. Крім того, для покращення розпізнавання природних мов платформи технологій машинного навчання пропонують більше можливостей для аналізу вигод користувачів, структурованого та контекстного управління діалогом користувачів та адаптивну підтримку декількох пристроїв для роботи на різних мовах. Найсучасніші платформи підтримуватимуть створення коду за допомогою штучного інтелекту, тому розробники можуть швидко додавати різноманітні сценарії використання. Це зменшить кількість коду, який потрібно писати та підтримувати.

Асистенти будуть не лише інформативними, але й активними. Сьогодні в більшості випадків помічники використовуються для отримання інформації та відповіді на запитання. Надалі ми побачимо, що віртуальні помічники зможуть не лише відповідати на запити користувачів, але й виконувати певні завдання самостійно. Завдяки інтегрованим платіжним системам та мережевим стандартам, таким як OAuth, помічники здійснюватимуть

платіжні операції, не перенаправляючи користувачів на інші інтерфейси. Наприклад, замовлення квитків, відправлення квітів, бронювання місць у ресторанах та багато іншого - все з голосовим керуванням на різних пристроях без необхідності відвідувати окрему програму чи веб-сайт.

Віртуальні помічники змінять наше сприйняття подорожі автомобілем. Сьогодні в машині їх використовують в основному для пошуку маршрутів та навігації, здійснення телефонних дзвінків, управління музикою, надсилання текстових повідомлень, тощо.

Екосистеми віртуальних помічників стануть більш відкритими та запропонують нові інструменти для природної взаємодії з користувачами. Це, насамперед, надання корисної інформації для водія у голосовому форматі та підтримка платіжних транзакцій.

Щороку американці витрачають загалом більше одного мільярда годин на шляху від дому до роботи. Оскільки відкриття веб-сайтів чи додатків під час руху небезпечно, розумний віртуальний помічник може взяти на себе деякі завдання користувача, забезпечуючи тим самим безпечніший трафік..

В майбутньому ми побачимо нові рішення, які перетворять віртуальних помічників із інструменту із вторинною функціональністю у повноцінну платформу споживчого інтерфейсу, не менш важливу, ніж інтерфейс мобільного пристрою.

Якщо проаналізувати найновіший цикл Gartner щодо нових технологій, можна побачити, що віртуальні помічники вже рухаються до провалу розчарувань, оскільки пройшли етап завищених очікувань. Складно визначити, як довго голосові помічники та розумні динаміки зможуть залишатися на вершині, перш ніж вони почнуть ковзати в яму розчарування.

У будь-якому випадку, голосові помічники ще мають пройти довгий шлях, перш ніж вони зможуть знайти своє місце на плато продуктивного використання. У той же час власники платформ шукатимуть схему створення

вигідної моделі для сторонніх партнерів, наприклад у формі додатку в магазині AppStore, щоб потужно просунути екосистему голосових технологій.

1.2 Аналоги проекту

Даний проект – голосовий асистент для пристроїв на базі операційної системи Android. Згідно з [2,3,4] Основними аналогами є голосові асистенти Аліса, Сірі, Алекса, Кортана та Асистент від компанії Google.

Аліса, Сірі, Алекс, Кортана - всі ці жіночі імена даються голосовим помічникам найбільших світових компаній. Єдине, що виділяється з цього списку - Google, який дав своєму помічнику назву “Помічник”.

За даними [5], Яндекс створив Алісу в 2017 році. Основна перевага цього голосового помічника порівняно з прямими конкурентами полягає в тому, що він працює на Android, iOS і навіть настільних комп'ютерах. Але Аліса ще не знає, як керувати розумними домашніми системами.

Проте їй керувати програмами легко (звичайно, з “Яндекс”). Вона просто викличе для вас таксі, покаже вам, як швидше та без заторів дістатись до роботи, включить потрібну музику та дасть вам знати, коли піде дощ.

Ви навіть можете поговорити з нею. “Аліса” вміє підтримувати розмову та жарт.

“Аліса” виступає в голосі російської дубляжної актриси Тетяни Шитової.

Багато рис особистості Аліси визначаються набором фраз, написаних редакторами Яндекс. Одним з авторів персонажа “Аліси” був журналіст і письменник Володимир Гур'єв. Однак розробники запевнюють, що “Аліса” не обмежена набором раніше прописаних відповідей: нейронна мережа асистента навчається на великому масиві російськомовних текстів, включаючи мережеві діалоги. Це вплинуло на характер програми: кілька користувачів стикнулись з відмовою “Аліси” давати відповіді на питання або з її грубістю. Розробники “Аліси” постійно стежать за її поведінкою і виправляють її.

“Аліса” здатна реагувати емоційно: наприклад, залежно від контексту, вона може бути веселою або сумною.

Google Assistant з'явився в травні 2016 року як продовження Google Now. Цей голосовий помічник безособовий - у нього немає імені. Він просто “помічник”.

Але якщо це “Помічник” від найбільшої в світі пошукової системи, ви здивуєтеся, наскільки він знає про вас. Він має доступ до історії відвідувань, геоданих та навіть дзвінків. Але не бійтеся: завдяки зібраним даним він зможе запропонувати вам відповідний товар, показати вам шлях до потрібної точки, а також розповість про плани в календарі.

Іноді він надмірно активний: він показує непотрібні новини та будує маршрути до місць, де ти вже був. Крім того, “Помічник” не допомагає керувати розумним будинком.

Siri, один з перших голосових помічників на ринку, з'явився в серпні 2011 року. Siri працює лише на пристроях Apple і має широкий спектр функцій. Наприклад, ви можете вільно розмовляти з нею - вона розуміє людську мову, а набір дотепних відповідей помічника розширюється з кожним роком.

Новини, погода, фільми, телепрограми - все це йому підпорядковане. Крім того, він має можливість контролювати деякі елементи розумного будинку.

Однак деякі користувачі відзначають, що продуктивність Siri іноді не дуже швидка порівняно з іншими голосовими помічниками. Інтеграція з іншими послугами також обмежується. Можливо, розробники вважають, що Siri вже досить розумна.

У листопаді 2014 року Amazon представила світові свого нового віртуального помічника - Alexa. Основна відмінність “Алекса” від інших подібних продуктів - вона “заточена” під контролем систем розумного дому. Alexa вміє керувати пилососами, освітленням та багатьма іншими функціями.

Раніше Alexa була зосереджена лише на розумному будинку, але тепер додаток-помічник також доступний на смартфонах. Такі параметри, як моніторинг погоди, новини та пробки, є для неї вже стандартними. Крім того, Amazon представила API відкритого помічника, який дозволяє Alexa працювати з будь-яким іншим додатком чи сервісом. Це дуже зручно замовити щось у інтернет-магазинів.

Ви також можете поговорити з нею, але брендинг часто спливає в розмовах, і Алекса пропонує вам купити щось на Amazon.

Перша демонстрація голосового помічника Кортана Майкрософт відбулася в квітні 2014 року. Сьогодні вона чудово почувається не тільки на платформі Windows, але і на Android і навіть iOS згідно з [6].

Ви можете вільно спілкуватися з Кортаною, оскільки її мова сповнена жартів. Ще одна перевага полягає в тому, що користувач може налаштувати, які дані передавати в Кортану.

Але якщо ви хочете знайти щось завдяки цьому віртуальному помічнику, ви знайдете це в пошуковій системі Bing. Крім того, версії iOS та Android мають менше функцій, ніж версія Windows. І він може керувати системами розумного будинку лише тоді, коли він уже встановлений на пристроях.

Особистий помічник Cortana створений для передбачення потреб користувачів. При бажанні йому може бути наданий доступ до особистих даних, таких як електронна адреса, адресна книга, історія веб-пошуку тощо - всі ці дані, які він використовує для прогнозування потреб користувача. Кортана замінює стандартну пошукову систему і її можна викликати натисканням кнопки “Пошук”. Ви можете ввести необхідний запит вручну або голосом. Вона знаходить необхідну інформацію на основі результатів пошуку в системі Bing, Foursquare та серед особистих файлів користувача. Також віртуальний помічник не без почуття гумору: вона може підтримувати розмову, співати пісні та розповідати жарти. Це заздалегідь нагадує про заплановану зустріч, день народження друга та інші важливі події. Наприклад, Кортана повідомляє,

якщо рейс користувача було скасовано або є пробки. Його інтерфейс має дуже гнучкі налаштування конфіденційності, які дозволяють користувачеві визначити, яку інформацію надавати віртуального помічника. На думку розробників, ні Siri, ні GoogleNow не можуть похвалитися таким рівнем контролю.

Cortana інтегрується із деякими програмами з Windows PhoneStore.

У Кортані буде встановлено вікове обмеження - користувачі, облікові записи Microsoft яких не досягли 13 років, не можуть користуватися послугами помічника. Коли ви спробуєте активувати помічницю і задати їй питання, власник почує: “Вибачте, ви повинні бути трохи старше, перш ніж я можу вам допомогти”. Це може бути зв’язано з тим, що велика кількість онлайн-сервісів, яких використовує Cortana, не призначені для користувачів молодшого шкільного віку та дітей.

Голосовий помічник Cortana інтегрований у Windows 10. Він не діє як окрема програма, а інтегрується в пошук Windows 10. Таким чином, активація Cortana відбувається, коли ви переходите до пошуку. За функціональністю він схожий на Windows Phone 8.1. Користувач може шукати інформацію за допомогою клавіатури та миші, а також голосом. Після розпізнавання команди помічник почне пошук інформації в мережі або на комп'ютерних накопичувачах. Кортана відсутня в Windows 10 Corporate LTSB.

Кортана збирає користувачів на комп’ютерах та надсилає в Microsoft контактну інформацію, інформацію про дзвінки та текстові повідомлення, історію відвідувань сайту. Для своєї роботи Кортана також вимагає, щоб операційна система дозволяла збирати та надсилати статистику про всю інформацію, що вводиться в комп'ютер (за допомогою розпізнавання голосу, рукописного тексту та використання клавіатури).

Оскільки проект більш комерційний, то основними аналогами його є веб-додаток та телеграм-бот для роботи з розкладом університету. Вони можуть дати тільки базову інформацію в зручному візуальному представлені. Це їх

достаток і не достаток водночас, так як містить зручний користувацький інтерфейс, дає відповідь на базові запитання, але не має функціоналу для обробки більш широкого спектру складності запитів, наприклад зібрати дані про всі навчальні пари протягом деякого проміжку часу деякого факультету чи факультетів.

По своїй суті проект об’єднує дві складові: розклад університету та голосовий асистент, і компонує їх у єдиний програмний продукт для мобільних пристроїв на базі операційної системи Android.

Веб-застосунок розроблений на замовлення приватним підприємством “ПОЛІТЕК-СОФТ” і має назву “ПС-розклад”, версія 3.8. Даний продукт містить доволі зручний користувацький інтерфейс та швидкий обробник запитів (рис. 1.1).

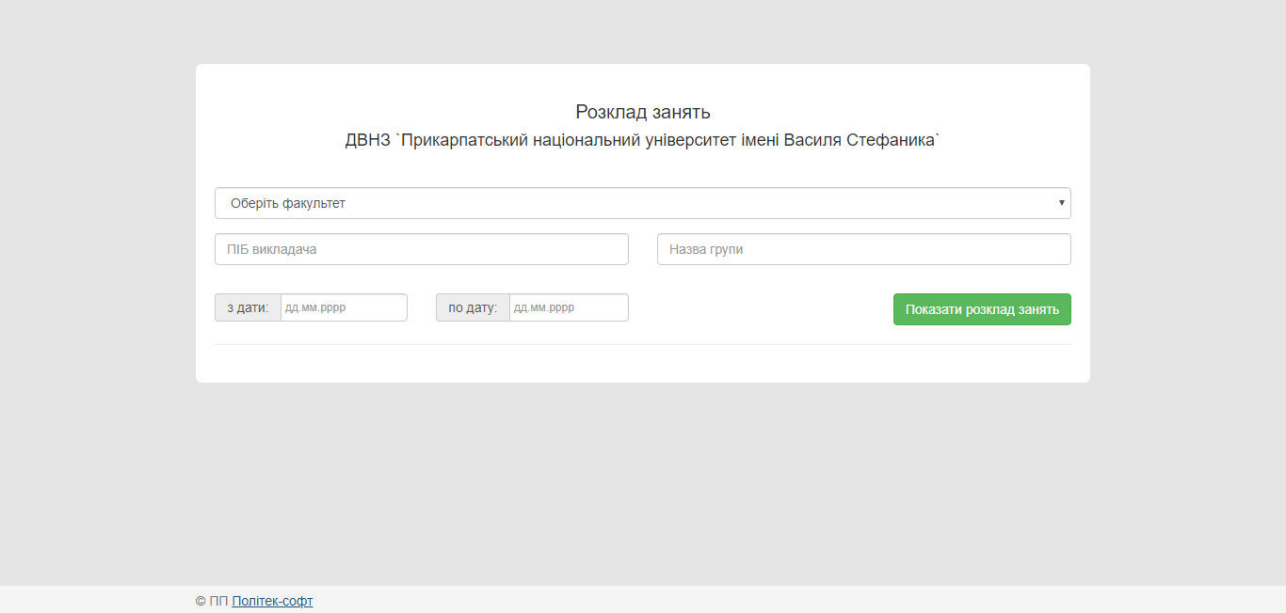

Рисунок 1.1 – Скріншот веб-застосунку для розкладу університету

Телеграм-бот розроблений безпосередньо студентами університету, містить зручний інтерфейс у вигляді діалогового вікна. Надає базову інформацію про розклад, проте цілком достатню для студента або викладача (рис. 1.2).

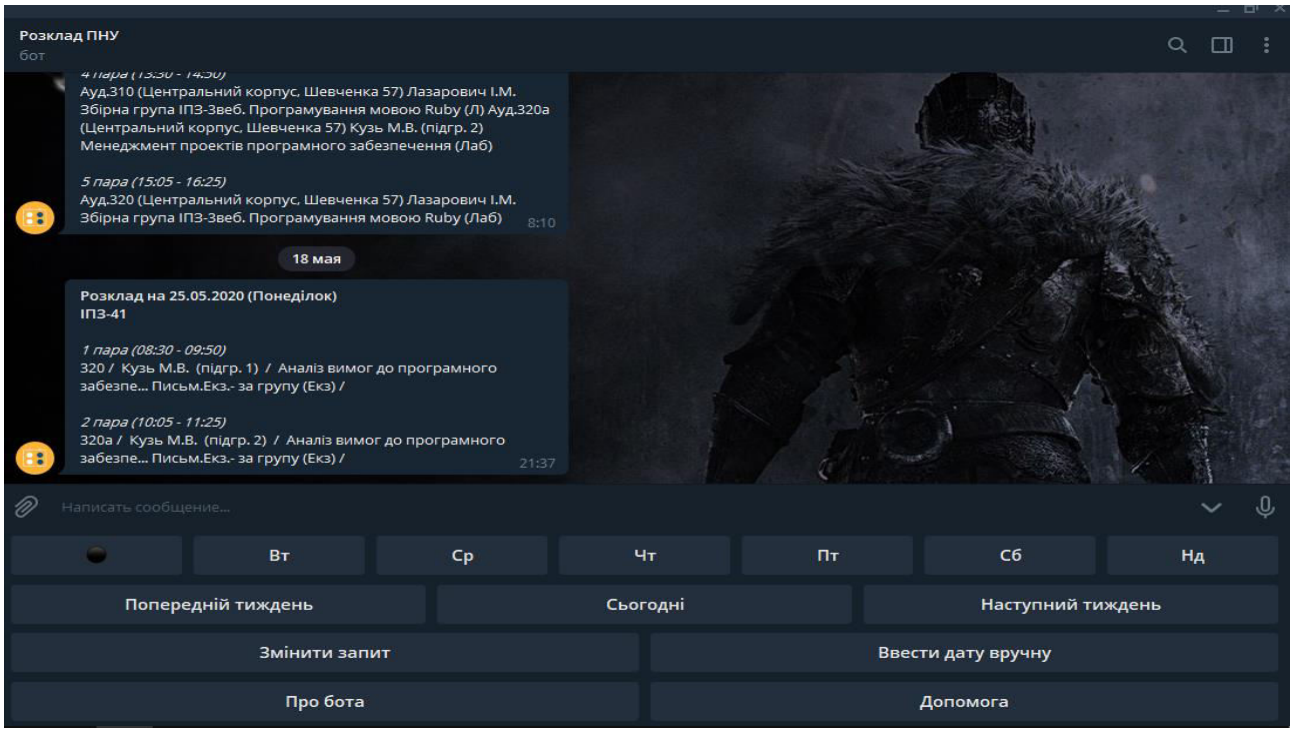

Рисунок 1.2 – Скріншот телеграм-боту для розкладу університету

1.3 Актуальність проекту

Проект даної дипломної роботи матиме назву “Dekanat” і призначений для роботи з розкладом навчальних пар Прикарпатського Національного Університету імені Василя Стефаника.

Актуальність даного голосового асистенту доволі велика для університету, так як призначена полегшити будні дні студентам та викладачам в межах вищого навчального закладу.

Доволі часто буває, коли відсутність доступу до розкладу чи його подача у вигляді таблиці на стенді, веб-додатку, телеграм-бота не дає студентам, викладачам, і не тільки, отримати потрібну інформацію вже і зараз.

Наприклад, на факультеті кілька десятків груп, кожна з яких має свій розклад навчальних пар. Факультет має обмежену кількість аудиторних кабінетів для проведення навчального процесу. Тому в розкладі враховується дана ситуація і кожна група займає різні аудиторії у різні проміжки часу. Але

часто бувають форс-мажорні обставини при яких більше одної групи займають ту ж саму аудиторію в однаковий проміжок часу, а деякі аудиторії порожні.

Така ситуація і не тільки призводить до запізнень на навчальні пари студентів, викладачів, відхилення від графіку навчального плану та інших. Оскільки факультетів в університеті кілька десятків, а груп більше тисячі – масштаб форс-мажорних обставин зашкалює.

Даний додаток призначений для зменшення масштабу форс-мажорних обставин.

Сьогодні більшість персоналу університету використовує смартфони на базі операційної системи Android. Тому було вирішено розробити проект у вигляді Android-додатку.

Архітектура додатку повинна дозволити студентам та викладачам з допомогою слів отримати більш складну і структуризовану інформацію із розкладу, наприклад список вільних аудиторій у даний проміжок часу, чи навіть забронювати вільну аудиторію. Крім цього можна буде отримати банальні дані з розкладу, наприклад присутність викладача на сьогоднішній день, назву аудиторії де згідно розкладу повинна відбутись навчальна пара, назву предмета, викладача, який її проводить, чи групу якій потрібно провести навчальну пару в даний момент та інші.

Однак з присутністю аналогів у вигляді веб-застосунку та телеграм-боту, існує імовірність низької актуальності проекту, точніше його реалізації у вигляді додатку з голосовим асистентом від компанії Google, оскільки не всі бажаючі зможуть використовувати дану частину проекту і не кожному доцільним буде використовувати її доволі часто. Існує імовірність що Android додаток буде заміненим веб-застосунком або телеграм-ботом в угоду зручності використання та універсальності для більшості доступних пристроїв.

2 ДИЗАЙН ТА АРХІТЕКТУРА ПРОЕКТУ

2.1 Загальна картина проекту

Даний проект – мобільний додаток для пристроїв на базі операційної системи Android. Додаток матиме назву “Dekan.at” і в якості іконки отримає логотип Прикарпатського національного університету імені Василя Стефаника (рис. 2.1).

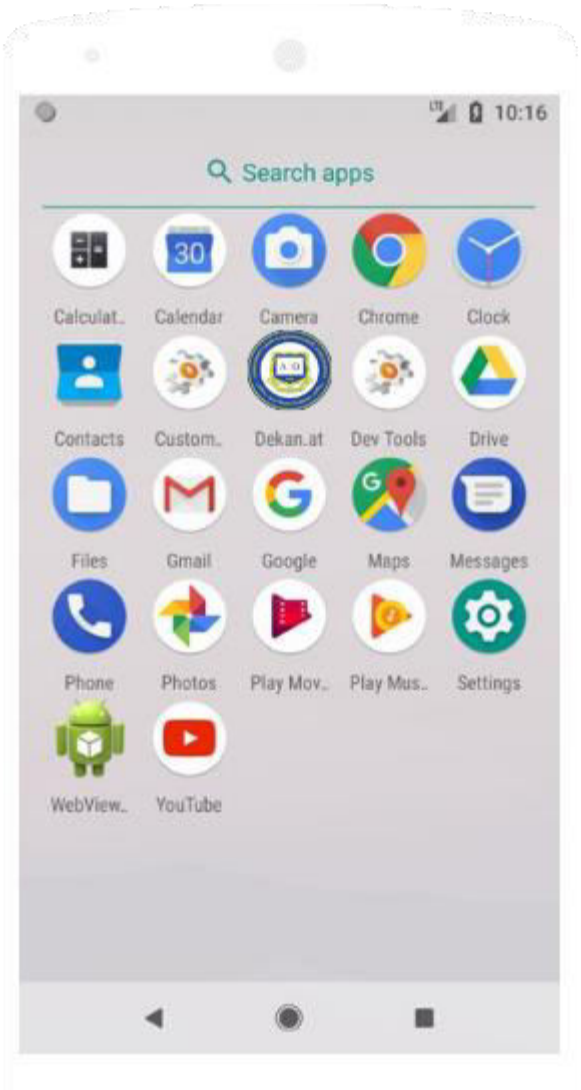

Рисунок 2.1 – Іконка додатку (нижче іконки камери)

Після тривалого часу розробки прототипу було вирішено, що проект міститиме простий для користувача інтерфейс, а функціонал буде поділений

на Front-end та Back-endчастини. Інтерфейс користувача міститиме кнопку для голосових команд, текстове поле, куди буде вводитися сказаний користувачем текст і пустий блок, у якому буде показуватись результат після обробки сказаного тексту (рис. 2.2).

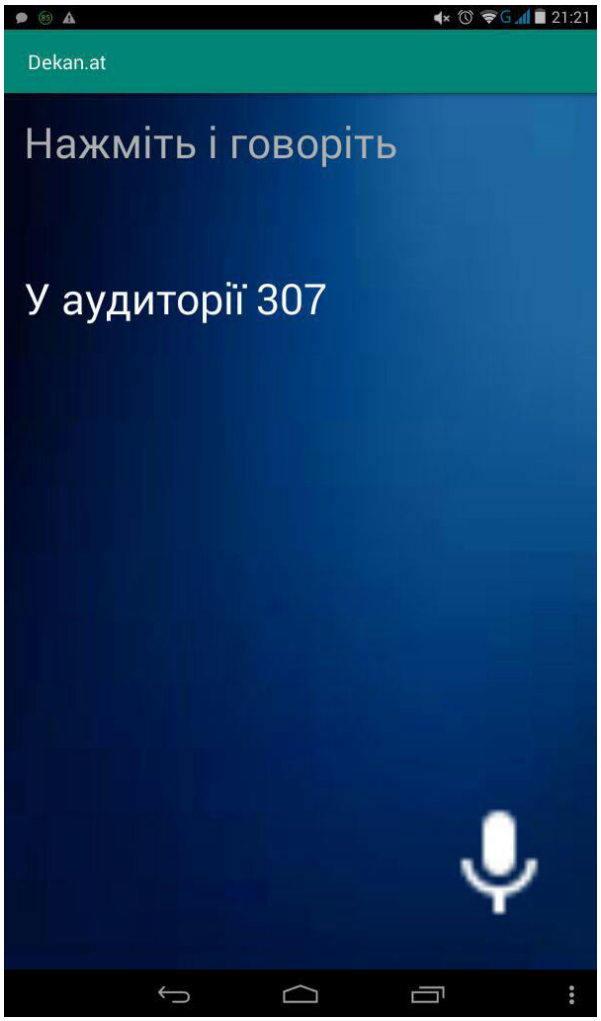

Рисунок 2.2 – Прототип додатку (результат обробки запиту: “де пара?”)

Перевагами проекту будуть зручність використання і висока швидкодія функціоналу, недолік – потрібність доступу до інтернету.

2.2 Дизайн та архітектура Front-endчастини

Front-end частиною даного проекту являється Android-додаток, названий “Dekan.at”. Його архітектура доведена до мінімалізму і дозволяє

користувачеві тільки авторизуватись та безпосередньо подавати голосові команди (рис. 2.3).

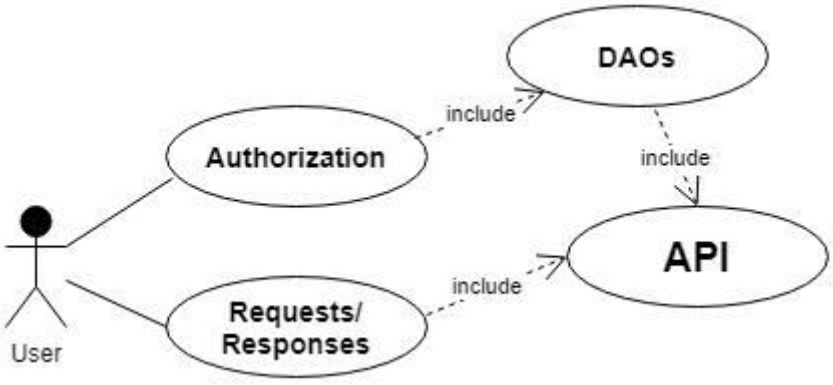

Рисунок 2.3 – UML-діаграма Front-endчастини

На даній діаграмі показано, що користувач, зайшовши у додаток, має обмежений набір функцій – авторизація та відправка запитів і отримання відповідей.

У свою чергу авторизація включає в себе DAO – DatabaseAccessObject – інструмент інтегрованого середовища розробки AndroidStudio для роботи з локальними базами даних SQLite. DAOвключає в себе певнідані з Back-endчастини проекту, а саме API. Це означає, що частина даних з API буде вноситись у базу даних без відома користувача – він отримає готовий робочий інтерфейс та функціонал для авторизації в додатку.

Основний функціонал для користувача, а саме подача запитів та отримання результату, також включає в себе функціонал API, оскільки усі внутрішні запити надходять саме туди. Користувачеві можливість подавати запити до API подається у вигляді голосового ассистенту. Із всього сказанного буде вибиратись певний набір строк, при якому відбудеться відправлення певного запиту. У відповідь API повертає результат у вигляді JSON об’єкту, над яким обробник виконує функцію парсингу та виводить отриманий результат на екран у вигляді тексту.

Після тривалої розробки, від концепції DAO було вирішено відмовидись в угоду швидкодії і меншої затратності ресурсів пристрою, крім цього авторизація була замінена на двокрокову (рис. 2.4).

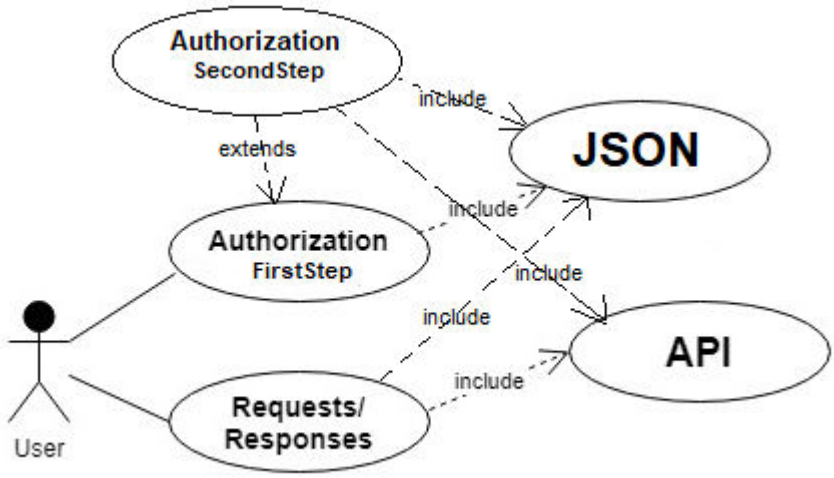

Рисунок 2.4 – UML-діаграма нової концепції front-end частини.

На даній діаграмі показано взаємодію користувача з компонентами front-end частини проекту та їх взаємодії між собою. Нова концепція від попередньої відрізняється сутністю, яка призначена для обробки локальних даних, та двох кроковою авторизацією. Сутність JSON призначена для роботи з локальними даними, тобто даними які повинні записуватись, зчитуватись, оновлюватись, видалятись виключно на Android пристрої.

JSON - це текстовий формат для обміну даними між комп'ютерами. JSON - читаний людиною текст. Формат дозволяє описувати об'єкти та інші структури даних. Цей формат в основному використовується для передачі структурованої інформації по мережі (через процес, який називається серіалізацією).

Формат був розроблений та популяризований Дугласом Крекфордом.

JSON знайшов свій сенс у розробці веб-застосунків при використанні AJAX технології. При використанні AJAX JSON під час асинхронного запиту виконує заміну XML. Перевага JSON перед XML полягає в тому, що він дозволяє складним структурам в атрибутах, займає менше місця і

безпосередньо з допомогою мови програмування JavaScript інтерпретується в об'єкти.

JSON з'явився у зв’язку необхідності спілкування з сервером у режимі реального часу без використання плагінів для браузерів, флеш-додатків або Java-аплетів, які використовувалися скрізь на початку 2000-х.

Дуглас Крекфорд був тим, хто активно просував новий формат у той час. Він та його колеги хотіли створити технологію, яка б використовувала можливості звичайного браузера, щоб веб-розробники могли створювати веб-додатки з постійним зв’язком із сервером. JSON був перший раз використаним у проекті Communities.com для CartoonNetwork, він дозволяв одночасно обмінюватися повідомленнями та керувати елементами DHTML.

Веб-сайт JSON.org був запущений у 2002 році. У грудні 2005 року Yahoo почав мігрувати деякі свої веб-сервіси на JSON. Google почав працювати з даною технологією в грудні 2006 року у власному протоколі GData.

Завдяки своїй стислості порівняно з форматом XML, JSON більш придатний для серіалізації доволі складних структур.

Кажучи про додатки у вебі, JSON таким чином є доречний у завданнях обміну даними між браузером та сервером (AJAX) та загалом між серверами (через інтерфейси HTTP). Даний формат ще й добре підходить для утримання більш складних та динамічних структур у базах даних реляційного типу або кешах файлів (рис. 2.5).

```
var ajaxData = '{"name": "wiki", "fname": "pedia", "rates": [1, 4, 5, 6]}'
var ajaxObj = JSON.parse(ajaxData)

alert(ajaxObj.name + ajaxObj.rates[2]) /* Виведе «wiki5» */
```

Рисунок 2.5 – Приклад використання JSON

JSON базується на таких структурах:

– Набір ім’я / значення пар. На різних мовах він реалізується як асоціативний масив, об'єкт, структура, словник, запис, хеш-таблиця або список з ключем;
– упорядкований список значень. У багатьох мовах це реалізується як послідовність, масив, вектор або список.

Це все є універсальними структурами даних. Усі сучасні мови програмування теоретично підтримують їх у різних формах. Так як JSON використовують для операцій передачі та зчитування даних і їх обміну між принципово різними між собою мовами програмування, є сенс будувати їх на цих структурах.

JSON використовує такі форми:

– об'єкт – це послідовність пар імен / значень. Об'єкт починається символом “{” і закінчується символом “}”. Кожне значення супроводжується символом “:”, а пари імен / значень розділяються таким символом як кома;
– масив є послідовністю різних значень. Починається із символу “[” та завершується символом “]”. Його значення розділені комами;
– значенням може бути рядок у подвійних лапках, або число, або логічне істинне чи помилкове, або нульове, або об'єкт, або масив. Ці структури можуть вкладатись;
– рядок - це послідовність нульових або більше символів Unicode, обмежених подвійними лапки, використовуючи послідовності втечі, що починаються із зворотної косої риски “\”. Символи представлені простим рядком.

Тип String (Рядок) дуже схожий на String у C та Java. Цифра також дуже схожа на номер C або Java, за винятком того, що не використовуються восьмеричний та шістнадцятковий формати. Проміжки можна вставити між будь-якими двома лексемами.

Наступний приклад показує JSON-представлення об'єкта, який описує людину. Об'єкт містить рядкові поля імені, об'єкт, який описує адресу, і масив, що містить список телефонів (рис. 2.6).

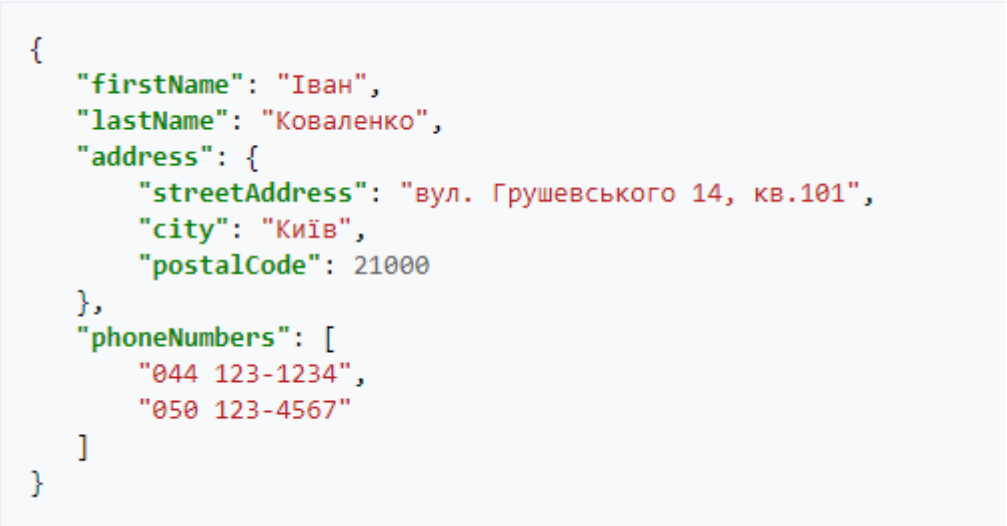

Рисунок 2.6 – JSONпредставлення об'єкта

Наведений нижче фрагмент коду JavaScript показує, як клієнт може використовувати XMLHttpRequest для запиту об’єктів JSON з сервера. Серверна частина коду пропускається, вона просто повертає рядок URL у форматі JSON за запитом (рис. 2.7).

```
var the_object;
var http_request = new XMLHttpRequest();
http_request.open( "GET", url, true );
http_request.send(null);
http_request.onreadystatechange = function () {
    if ( http_request.readyState == 4 ) {
        if ( http_request.status == 200 ) {
            the_object = JSON.parse(http_request.responseText);
        } else {
            alert( "There was a problem with the URL." );
        }
        http_request = null;
    }
};
```

Рисунок 2.7 – Використання JSON в AJAX

Слід зазначити, що використання XMLHttpRequest не є крос-браузером, і код буде дещо змінений в різних версіях Internet Explorer, Edge, Opera, Safari або Mozilla. Використання запиту XMLHttpRequest обмежено одним правилом джерела: URL-адреса, яка відповідає на запит, повинна посилатися на той самий сайт, що обслуговував сторінку, що ініціювала запит.

Браузери також можуть використовувати тег <iframe> для асинхронного запиту даних JSON у крос-браузерній версії або використовувати просте пере направлення: “ <form action = “url_to_script” target = “name_of_hidden_iframe” >”. Цей підхід був поширеним до появи популярного на сьогодні XMLHttpRequest запиту.

Ви також можете використовувати динамічний тег <script> для завантаження даних JSON. Цей прийом можна обійти занадто суворим правилом єдиного джерела, але це не є безпечним. Запит JSONRequest пропонується як більш безпечна альтернатива.

Хоча JSON призначений для серіалізованої передачі даних, його синтаксис відповідає синтаксису JavaScript, і це створює ряд проблем безпеки. Часто функція eval () використовується для обробки даних, отриманих із зовнішнього джерела у форматі JSON, без попередньої перевірки.

Оскільки JSON - синтаксично правильний фрагмент коду JavaScript, природним способом розбору даних JSON в програмі JavaScript є використання вбудованої функції JavaScript eval (), яка призначена для обчислення виразів JavaScript. Цей підхід виключає потребу в додаткових аналізаторах.

Техніка використання eval () робить систему вразливою, якщо джерело використаних даних JSON не є надійним. Такі дані можуть бути зловмисним кодом JavaScript для атак введення коду. За допомогою цієї вразливості можна вкрасти дані, підробити автентифікацію. Однак вразливість може бути усунена за допомогою додаткових інструментів перевірки даних. Наприклад, перед запуском eval () дані з зовнішнього джерела можна перевірити, використовуючи регулярні вирази. RFC, що визначає JSON, рекомендує використовувати алгоритм, наведений нижче, для перевірки джерела на відповідність формату JSON (рис.2.8).

```
const my_JSON_object = !(/[^,:{}\[\]0-9.\-+Eaeflnr-u \n\r\t]/.test(
text.replace(/"(\\.|[^"\\])*"/g, '''))) &&
eval('(' + testedData + ')');
```

Рисунок 2.8 – Використання функції eval()

В якості більш безпечної альтернативи eval () запропонована нова функція parseJSON (), яка може обробляти лише дані JSON. Він був представлений у четвертій версії стандарту ECMAScript і описаний у статті під назвою “JSON: нежирна альтернатива XML”.

Функцію JSON.parse можна використовувати для розбору JSON.

Підтримка JSON вбудована в останніх версіях браузерів, тобто вони здатні обробляти його, не викликаючи функцію eval (), що призводить до описаної проблеми. Обробка JSON в цьому випадку зазвичай швидша. Так у червні 2009 року такі браузери, як FireFox 3.5, Internet Explorer 8, Opera 10.5, Chrome, Safari мали вбудовану підтримку JSON.

Принаймні дві популярні бібліотеки JavaScript використовують вбудований JSON, коли він доступний: jQuery та Dojo.

Неправильне використання JSON робить сайти вразливими до підробки міжпрофільних запитів (CSRF або XSRF). Оскільки тег <script> дозволяє використовувати джерело, яке не належить до того ж домену, що і використовуваний ресурс, воно дозволяє обробити та виконати код представлених у JSON форматі даних у контексті будь-якої сторінки, що дозволяє компрометувати паролі або інша конфіденційна інформація користувачів. отримали авторизацію на іншому сайті.

Це проблема лише в тому випадку, якщо дані JSON містять конфіденційну інформацію, яка може бути порушена третьою стороною, і якщо сервер спирається на єдину політику джерела, при виявленні запиту із зовні блокуючи доступ до даних. Це не проблема, якщо сервер визначає обґрунтованість запиту, надаючи дані лише в тому випадку, якщо він правильний. Для цього не можна використовувати файл cookie HTTP. Ексклюзивне використання файлу cookie HTTP використовується для підробки міжпрофільних запитів.

Для даного проекту сутність JSON включає в себе три складові: insertJsonToFile, getJsonFromFile, insertJsonToFileFromUrl – кожна з яких призначена для роботи з локальними JSON файлами (рис. 2.9).

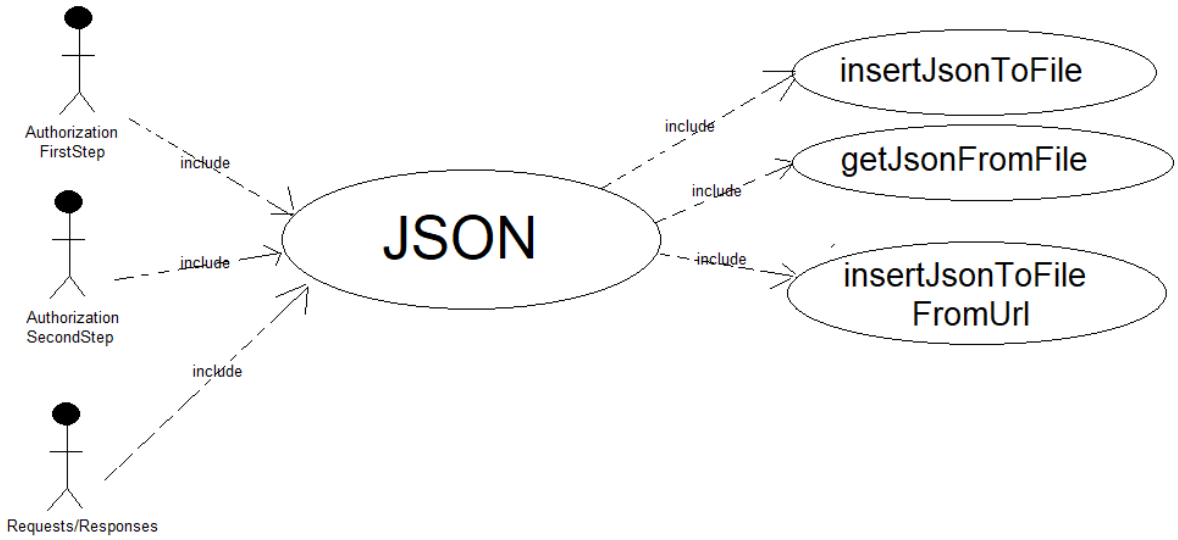

Рисунок 2.9 – UML-діаграма сутності JSON

Сутність JSON використовується такою складовою проекту як авторизація та “Request/Response”. Алгоритм роботи авторизації простий у розумінні, мало затратний для ресурсів пристрою на якому буде виконуватись, має доволі високу швидкодію (рис. 2.10).

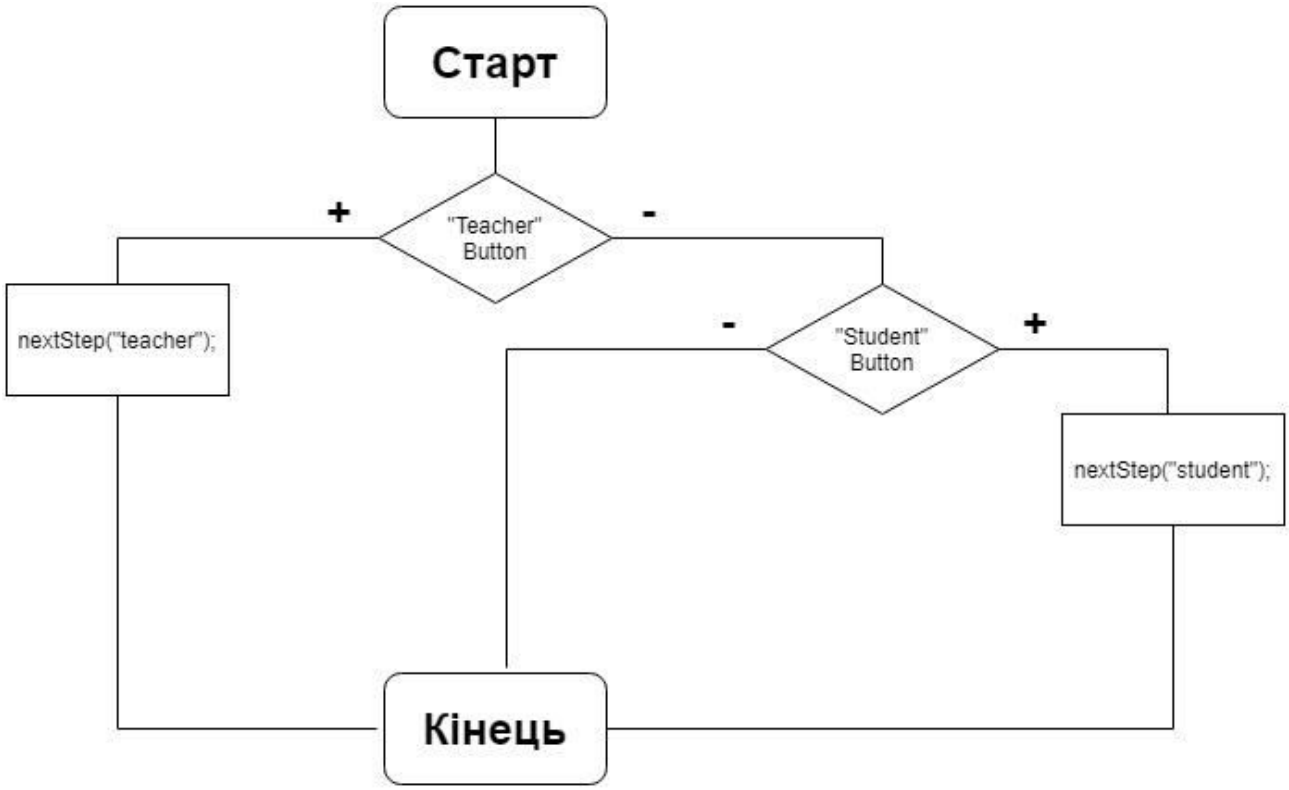

Рисунок 2.10 – Блок-схема сутності AuthorizationFirstStep

На даній блок-схемі показано перший крок авторизації: при натисканні на кнопку “Teacher” виконується функція nextStep() куди передається параметр у вигляді строки “teacher”, при натисканні на кнопку “Student” також виконується функція nextStep() але вона приймає параметр “student”. Суть функції nextStep() полягає у тому, що вона з допомогою об’єкту класу Intent переходить на вікно сутності AuthorizationSecondStep і передає туди отриманий нею параметр.

Суть другого кроку авторизації полягає у тому, що користувач після того як вказав чи являється він студентом або викладачем вибирає назву своєї навчальної групи або ж власне прізвище та ініціали, якщо це викладач, зі списку (рис. 2.11).

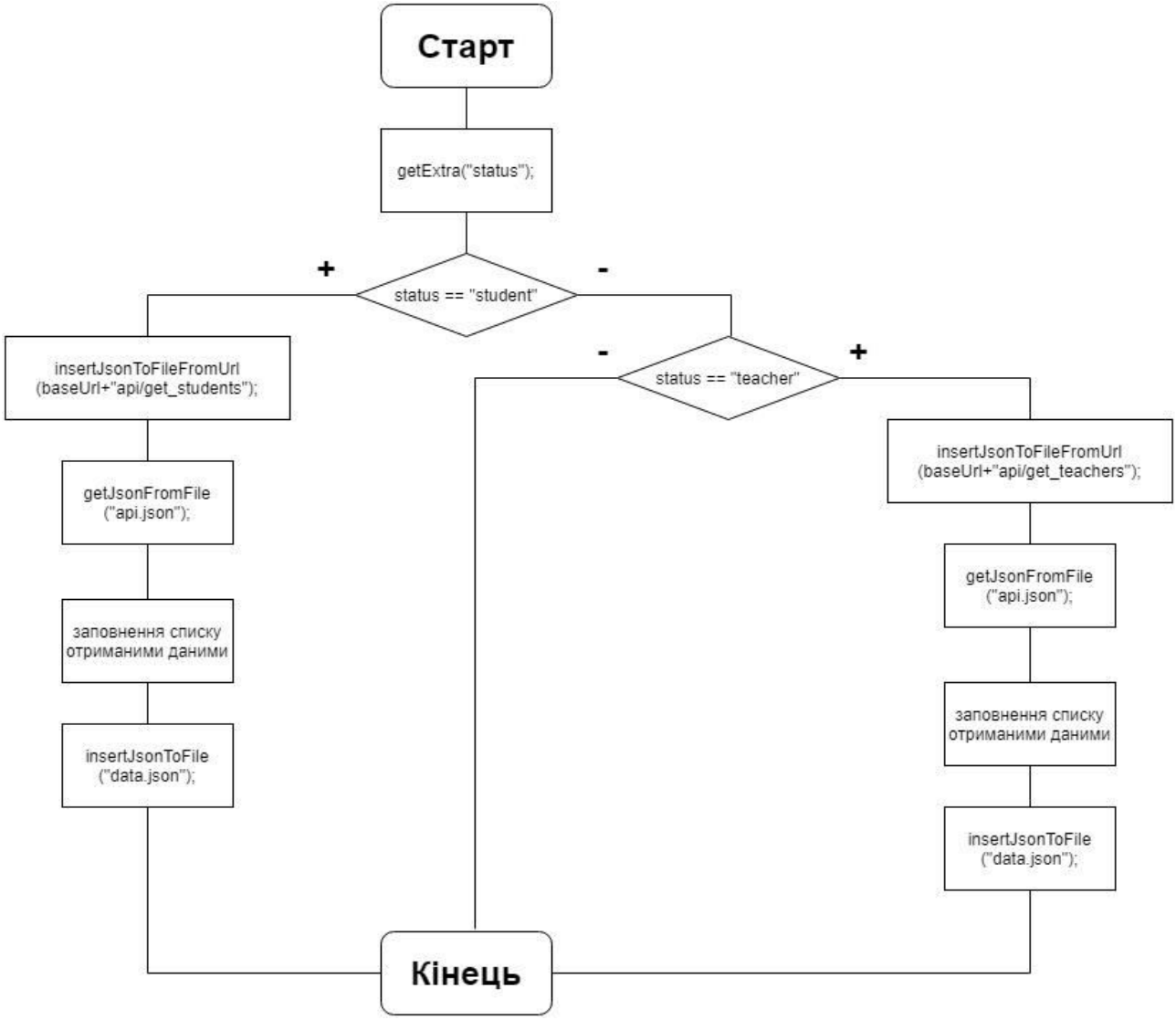

Рисунок 2.11 – Блок-схема сутності AuthorizationSecondStep

При другому кроці авторизації відбувається отримання статусу із об’єкту класу Intent методом getStringExtra() і в залежності від нього відправляється GET-запит на адресу back-end частини проекту – API під назвою DekanatAPI, де зміна baseUrl дорівнює адресі домену API даного проекту.

Intent (Намір) - це об’єкт обміну повідомленнями, який можна використовувати для запиту дії з компонента іншої програми. Хоча об'єкти наміру в декількох аспектах спрощують обмін даними між компонентами, вони в основному використовуються в трьох ситуаціях:

- почати операцію. Компонент Activity - це один екран у програмі. Щоб запустити новий примірник компонента Activity, потрібно передати об’єкт Intent методу startActivity (). Об'єкт Intent описує те, для чого потрібно запустити, та містить усі інші необхідні дані. Якщо ви хочете отримати результат після операції, зателефонуйте на метод startActivityForResult (). Ваша операція отримає результат у вигляді окремого об'єкта наміру в методі зворотного виклику операції onActivityResult ();
- почати послугу. Сервіс – це компонент, який виконує дії у фоновому режимі без користувальницького інтерфейсу. Ви можете запустити службу для одноразової дії (наприклад, для завантаження файлу), передавши об’єкт Intent методу startService (). Об'єкт Intent описує послугу, яку ви хочете запустити, і містить усі інші необхідні дані. Якщо служба розроблена за допомогою інтерфейсу клієнт-сервер, ви можете зв’язати її з іншим компонентом, передавши об’єкт наміру методу bindService ();
- для надсилання широкомовної інформації Поширене повідомлення - це повідомлення, яке може отримати будь-яка програма. Система видає різні широкомовні повідомлення про системні події, наприклад, коли система завантажується або пристрій починає заряджатися. Щоб надсилати широкомовні повідомлення іншим

програмам, ви повинні передати об’єкт Intent методу sendBroadcast (), sendOrderedBroadcast () або sendStickyBroadcast ().

Існують такі типи Intent:

- явні об'єкти Intent вказують компонент, який слід запускати по імені (повна назва класу). Явні об'єкти намірів, як правило, використовуються для запуску компонента з вашої власної програми, оскільки ви знаєте ім'я операційного класу або служби, які потрібно запустити. Наприклад, ви можете запустити нову операцію у відповідь на дії користувача або запустити службу для завантаження файлу у фоновому режимі;
- Неявні об'єкти Intent не містять імені конкретного компонента. Натомість вони зазвичай заявляють про вжиті дії, що дозволяє компоненту з іншої програми обробляти запит. Наприклад, якщо ви хочете показати користувачеві місце на карті, ви можете скористатися неявним об'єктом намірів, щоб попросити іншу програму зробити це, в якому ця функція надана..

Коли для запуску операції або послуги створюється явний об'єкт Intent, система негайно запускає компонент програми, визначений в об'єкті наміру.

Коли створюється неявний об'єкт Intent, Android знаходить відповідний компонент, порівнюючи вміст об'єкта Intent з фільтрами Intent, оголошеними у файлах маніфестів інших програм, доступних на пристрої. Якщо об'єкт Intent відповідає фільтру Intent, система запускає цей компонент і передає йому об'єкт Intent. Якщо знайдено кілька фільтрів намірів, система відображає діалогове вікно, де користувач може вибрати програму для додавання коментарів.

Фільтр Intent - це вираз у файлі маніфесту програми, який вказує типи об'єктів Намір, які може прийняти компонент. Наприклад, оголошуючи фільтр намірів для операції, ви дозволяєте іншим програмам запускати вашу операцію безпосередньо з об'єктом наміру. Аналогічно, якщо ви не оголосите жодних

фільтрів намірів для операції, її можна запустити лише за допомогою явного об'єкта Intent.

Домен або доменне ім’я - це частина ієрархічного простору імен в Інтернеті, яка обслуговується групою серверів системних імен (DNS) і централізовано керується.

DNS-сервери містять в собі дані про вузли, назви кожного з яких належать домену, і переводять їх імена в адреси. Кожен домен має унікальне ім’я, і кожен комп'ютер, підключений до Інтернету, зазвичай має ім'я домену. Домени мають ієрархічне відношення. Два домени, розташовані на сусідніх рівнях ієрархії, називаються доменами вищого та нижчого рівнів відповідно. Домени верхнього (верхнього) рівня можуть формуватися за організаційними або географічними ознаками. Домени, утворені за географічними ознаками, об'єднують вузли, що належать до певної держави. Географічно вони поєднують в основному комп'ютери, розташовані в США. Доменна зона - сукупність доменних імен певного рівня, що входять до певного домену.

Згідно з доповіддю Verisign Domain Brief, Verisign, найбільшого в світі реєстратора домену, наприкінці 2012 року кількість зареєстрованих доменних імен другого рівня в усьому світі зросла до 250 мільйонів, навіть перевищила цю кілікість. Доменна зона .com за кількістю реєстрацій є абсолютним лідером, на неї припадає 40% усіх доменів. Серед інших популярних зон реєстрації доменів Verisign розрізняє зони .net, .uk, .org, .info, .cn, .nl та .ru..

Ім’я домену може складатися лише з обмеженого набору символів ASCII, що дозволяє вводити доменну адресу незалежно від мови користувача. ICANN затвердив систему IDNA на основі Punycode, перетворивши будь-яку рядок в кодуванні Unicode в допустимий набір символів DNS.

Сутність “Request/Response” займається обробкою запитів від користувача до сервера та відповідей від сервера до користувача. Спочатку вона отримує дані з локального JSON файлу, при отриманні помилки відправляє на реєстрацію, де локальний файл буде згенерований, при успіху

обробляє голосову команду користувача, оприділяє його статус (студент або викладач) і в залежності від нього виконує функції для обробки команди (рис. 2.12).

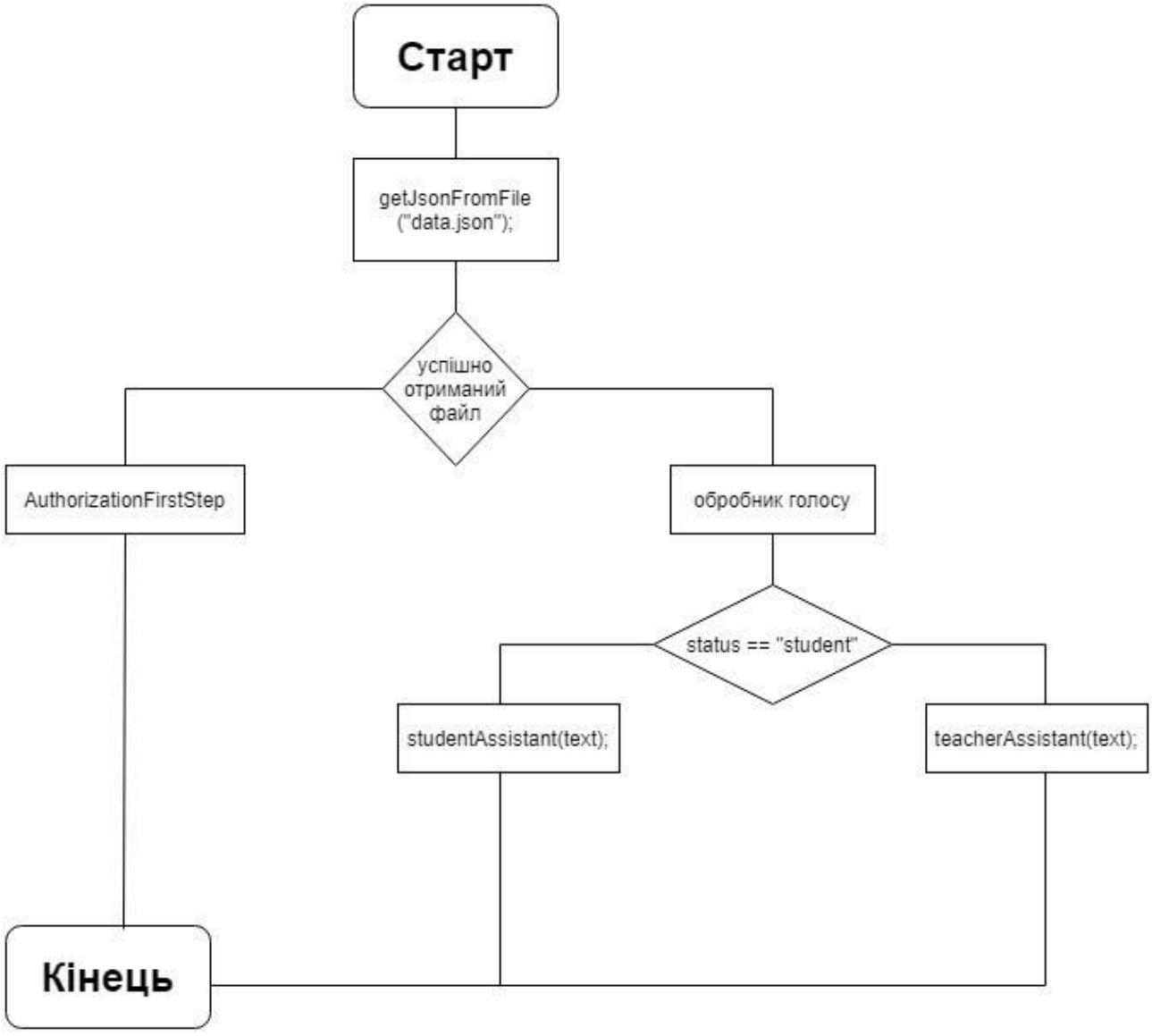

Рисунок 2.12 – Блок-схема сутності “Request/Response”

Функція getJsonFromFile призначена для отримання локального JSONфайлу. При помилці, вона повернее JSON об’єкт з ключем “error”, з допомогою якого відбувається перевірка на наявність файлу. При наявності файлу використовується обробник голосу від компанії Google. Після цього оприділяється статус користувача який отримується з локального JSONфайлу і

в залежності від нього виконується функція studentAssistant або teacherAssistant куди передається параметр text – текст, що утворився в результаті обробки голосової команди користувача. Ці функції відправляють спеціальні HTTP запити взалежності від отриманого тексту. Якщо команда асистенту невідома, він відправить сповіщення з допомогою об’єкта класу Toast.

Toast є спливаючим повідомленням з даними про деяку операцію. Він займає мало місця і при цьому не відволікає користувача від поточноих дій та не порушує роботу програми. Після завершення тайм-ауту Toast автоматично зникає.

Наприклад, якщо натиснути кнопку “Надіслати” в електронному листі, з'явиться Toast із повідомленням “Надсилання повідомлення ...” (рис. 2.13).

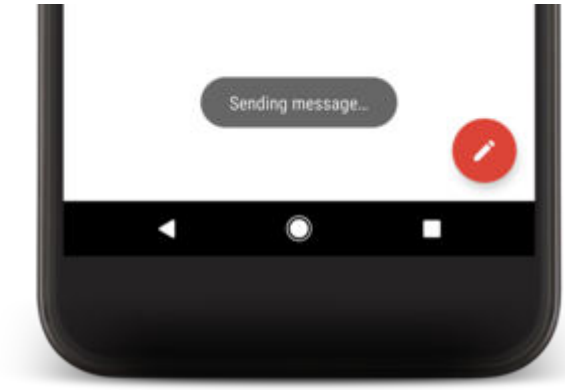

Рисунок 2.13 – Приклад відображення toast

2.3 Дизайн та архітектура Back-end частини

Back-end частина проекту – API, назване як “DekanatAPI” (рис. 2.14).

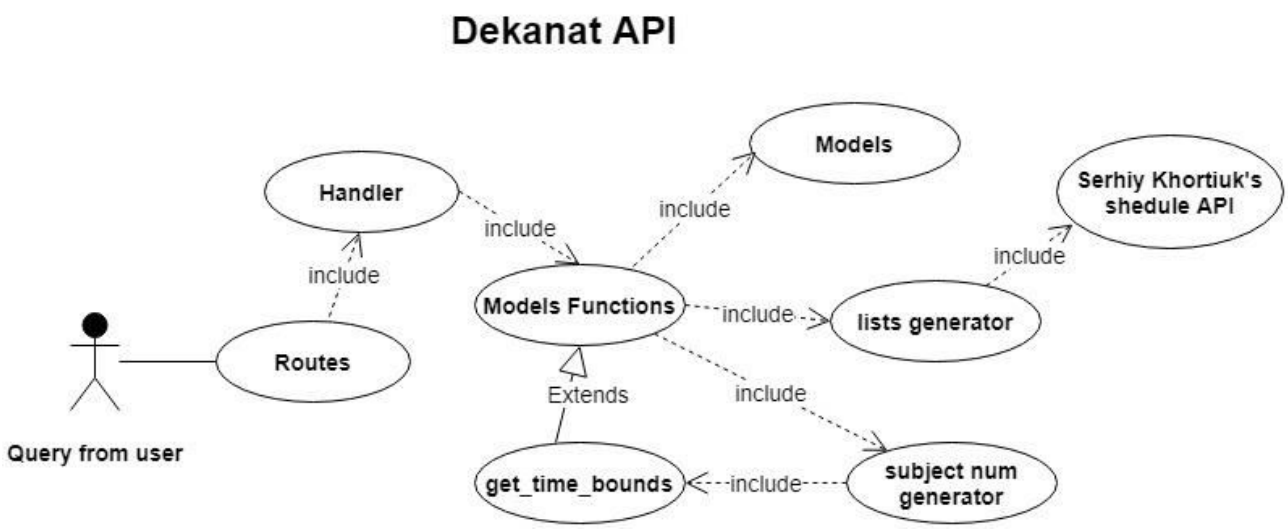

Рисунок 2.14 – UML-діаграма Back-endчастини

На діаграмі зображено як згенерований запит від користувача потрапляє до API. Для коректної роботи, APIскладається із таких сутностей:

- Routes;
- Handler;
- Models functions;
- Models;
- Lists generator;
- Schedule API;
- Subject num generator;
- Get time bounds.

Сутність Routesмістить так званні “Роути” – допустимі HTTP-запити у вигляді URL-адресів, перехід на яких повертає результат сутності Handler.

Сутність Handler – це обробник, який приймає дані з “роутів” і повертає результат роботи сутності Models Functions.

Models Functions – містить набір функцій, призначених для роботи з моделями бази даних.

Models – моделі бази даних.

Lists generator – генератор списків. Він отримує дані із ScheduleAPI, виконує над ними операцію парсингу. З отриманого результату генерує списки із записами, які будуть заноситись у власну базу даних.

Scheduler API – API яке повертає дані про розклад навчальних пар.

Subject num generator – генератор номеру навчальної пари. Отримує поточний час запиту користувача і зрівнює його з даними, одержаними з допомогою Gettimebounds. Повертає порядковий номер навчальної пари.

Gettimebounds – функція для стягнення даних про час проведення навчальних пар з бази даних даного API. Використовується генератором порядкового номеру навчальної пари.

API (Application Programming Interface) - набір визначень процедур, протоколів взаємодії та інструментів для створення програмного забезпечення. Спрощений - це набір чітко визначених методів взаємодії різних компонентів. API надає розробнику інструменти для прискорення процесу розробки програмного продукту. Воно застосовується для веб-систем, операційних систем, баз даних, апаратних засобів, бібліотек програмного забезпечення.

Однією з найпоширеніших цілей API є надання набору широко використовуваних функцій, таких як малювання вікна або піктограм на моніторі. Розробники програмного забезпечення користуються перевагами API у розробці функціоналу, тому їм не потрібно розробляти все з нуля. API - це абстрактне поняття - програмне забезпечення, яке пропонує певний API, часто називають реалізацією цього API. У багатьох випадках API є частиною набору для розробки програмного забезпечення, однак, комплект розробок може включати як API, так і інші інструменти / обладнання, тому два терміни не є взаємозамінними..

API високого рівня часто втрачають гнучкість. Виконувати деякі функції нижчого рівня стає набагато складніше або навіть неможливо.

Призначення даного API – полегшити розробку проекту. Усі складні процеси виконує сервер. Додаток являється способом подачі результату роботи Back-end частини проекту.

У загальному архітектура алгоритму даного API виглядає так: щогодини Dekanat API відправляє GET HTTP запит до Schedule API звідки дістає всі необхідні дані, після цього відбувається збір даних в асоціативний масив, значення кожного ключа якого присвоюється списком однакових за структурою але різними за значеннями даними. Пізніше дані списки використовуються для заповнення власної SQLite бази даних проекту та окремих функціоналів API та Android-додатку.

SQLite - це легка система управління реляційними базами даних. Втілена як бібліотека, де реалізовано багато стандартів SQL-92. Вихідний код SQLite поширюється як загальнодоступний домен (publicdomain), тобто його можна використовувати без обмежень і безкоштовно для будь-яких цілей. Фінансову підтримку розробників SQLite надає створений спеціальний консорціум. До нього входять компанії Oracle, Adobe, Nokia, Mozilla, Bloomberg та Bentley.

З 2018 року SQLite, як JSON та CSV, є рекомендованим форматом для зберігання структурованого набору даних бібліотекою Конгресу.

Станом на 2005 рік даний проект отримав Google-O'Reilly Open Source Awards.

Особливість SQLite полягає в тому, що він не використовує парадигму клієнт-сервер, тобто двигун SQLite не є окремим процесом, з яким програма взаємодіє, але надає бібліотеку, з якої складається програма і двигун стає частиною програми. Таким чином, виклики функцій бібліотеки SQLite (API) використовуються як протокол обміну. Такий підхід скорочує накладні витрати, час реагування та робить простішою програму. SQLite повністю зберігає в собі базу даних (враховуючи таблиці, індекси, визначення та дані) в єдиному стандартному файлі на комп'ютері, на якому працює програма.

Простота реалізації досягається завдяки тому, що перед транзакцією весь файл, який зберігає базу даних, блокується; Функції ACID досягаються, зокрема, створенням журнального файлу.

Кілька процесів або потоків можуть без проблем читати дані з однієї бази даних одночасно. Запис у базу даних можна здійснити лише в тому випадку, якщо зараз не обслуговуються інші запити; інакше спроба запису не вдається і код помилки повертається програмі. Інший варіант - автоматично повторити спроби запису протягом визначеного часового інтервалу.

В комплект також входить функціональна клієнтська частина у вигляді виконуваного файлу sqlite3, який демонструє реалізацію функціоналу власної бібліотеки. Частина зі сторони клієнта працює з допомогою інтерфейсу командного рядка та дозволяє отримати доступ до файлу бази даних на основі типових функцій ОС.

Завдяки архітектурі двигуна SQlite можна використовувати як у вбудованих системах, так і на спеціалізованих машинах з наборами даних гігабайт.

Особливостями SQLite є:

- транзакції навіть після збоїв системи і збоїв живлення лишаються атомарними, послідовними, ізольованими та міцними (ACID);
- встановлюється без конфігурацій, тобто не потребує установки та адміністрування;
- реалізовує доволі велику частину SQL92 стандарту;
- база даних міститься в одному файлі на диску, який є крос-платформенним;
- підтримує терабайтні розміри баз даних та гігабайтний розмір рядків а також і BLOBів;
- невеликий розмір коду: менше 350 КБ повністю налаштовано та менше 200 КБ із пропущеними додатковими функціями;

- швидший, ніж популярні двигуни бази даних клієнт-серверного типу, для найбільш поширених операцій;
- легкий та простий при використанні API;
- написана в кодуванні ANSI C, має включену прив'язку до TCL; також містить доступні прив'язки для багатьох різних мов;
- чудово прокоментовано сирцевий код з повністю протестованим покриттям гілок;
- є доступним в якості єдиного файлу, якого можна без великих зусиль вставити в той чи інший проект;
- автономний;
- крос-платформенний: підтримує операційні системи Linux, Mac OS, Windows. Без проблем може переноситись на інші платформи.

Створення та обслуговування баз даних може здійснюватися за допомогою текстової консолі з командами SQL або за допомогою спеціальних інструментів, включаючи графічний інтерфейс користувача.

Для коректної роботи API, і взагалі всього проекту, використовується власна база даних. Архітектура її моделей дозволяє виконувати складні запити, наприклад як дізнатись вільну аудиторію у даний момент часу, чи навіть забронювати вільну аудиторію (буде доступно тільки викладачам) (рис. 2.15).

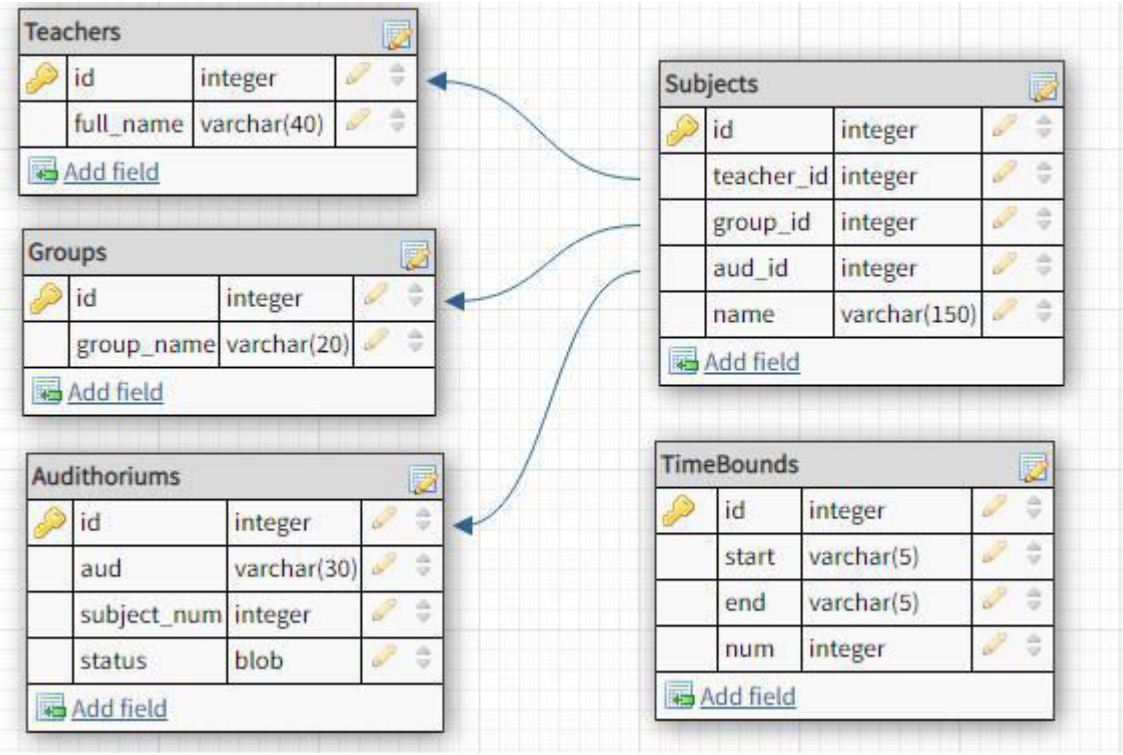

Рисунок 2.15 – Діаграма бази даних для API

Для оновлення даних API щогодини було розроблено Scheduler – програма яка відсилатиме спеціальні POST-запити по HTTP протоколу, призначення яких заключається в тому, що при їх виконанні DekanatAPI подаватиме запит до SchedulerAPI після чого генеруватиме списки потрібних даних і вноситиме їх у базу даних.

3 ПРАКТИЧНА РЕАЛІЗАЦІЯ ПРОЕКТУ

3.1 Розробка Front-end частини

Front-end частина проекту розроблена мовою програмування Java для операційної системи Android.

Java - об'єктно-орієнтована мова програмування, випущена в 1995 році компанією Sun Microsystems як основний компонент платформи Java. Oracle працює над мовою з 2009 року, придбавши Sun Microsystems у тому ж році. При офіційній реалізації програми Java складаються в байт-код, який інтерпретується віртуальною машиною для певної операційної системи.

Компанія Oracle надає віртуальну машину Java та компілятор Java та, які відповідають специфікаціям Процесу спільноти Java згідно Загальної публічної ліцензії GNU.

Мова запозичила багато синтаксису з C і C ++. Зокрема, за основу взята об'єктна модель C ++, але вона модифікована. Можливість виникнення деяких конфліктних ситуацій, які можуть виникнути через помилки програміста, було ліквідовано та полегшено процес розробки об'єктно-орієнтованих програм. Ряд дій, які програмісти повинні виконати на C / C ++, призначається віртуальній машині. Java розроблена насамперед як незалежна від платформи мова, тому має менші апаратні можливості низького рівня, що знижує швидкість роботи додатків порівняно, наприклад, з C ++. При необхідності Java дозволяє викликати написані на інших мовах програмування підпрограми.

Java мала вплив на розробку J ++, розроблену Microsoft. Робота над J ++ була припинена через позов, поданий Sun Microsystems, оскільки мова програмування була модифікацією Java. Пізніше на новій платформі Microsoft .NET було випущено J # для полегшення міграції J ++ або Java розробників на нову платформу. Згодом нова мова програмування C # стала основною мовою платформи, яка перейняла більшу частину Java. J # востаннє включався в

Microsoft Visual Studio 2005. Мова сценаріїв JavaScript має схоже ім'я Java та синтаксис, але не пов'язана з Java.

Мова спочатку називалася Дуб і була розроблена Джеймсом Гослінг для програмування електронних пристроїв споживачів. Пізніше він був перейменований в Java і використовувався для написання клієнтських програм та серверного програмного забезпечення. Він названий на честь торгової марки Java, яка, в свою чергу, була названа однойменним островом (Java), тому офіційна емблема мови зображує чашку з паркою кави. Існує ще одна версія походження назви мови, пов’язана з алюзією на кавову машину як приклад побутового приладу, для програмування якої мова була створена спочатку.

П’ять початкових цілей було при створенні мови програмування Java:

- синтаксис мусить бути об'єктно-орієнтовним, простим та звичним;
- безвідмовною та безпечною повинна бути реалізація;
- переносність та незалежність від архітектури повинні зберегтися;
- висока продуктивність при виконанні;
- мова програмування повинна бути із динамічним зв'язуванням модулів та інтерпретованою,.

“Незалежна від архітектури” означає, що програма, написана на Java, буде працювати на будь-якій підтримуваній апаратній або системній платформі без змін вихідного коду та рекомпіляції.

Цього можна досягти, компілювавши вихідний код Java в байт-код, що є спрощеною машинною інструкцією. Потім програма може бути виконана на будь-якій платформі, на якій встановлена віртуальна машина Java, яка інтерпретує байт-код у код, адаптований до специфіки конкретної операційної системи та процесора. Віртуальні машини Java зараз існують для більшості процесорів та операційних систем.

Стандартні бібліотеки надають звичайний спосіб доступу до функцій, залежних від платформи, таких як обробка графіки, багатопотоковість та

мережа. У деяких версіях для підвищення продуктивності JVM байт-код може бути складений в машинний код до або під час виконання програми.

Основною перевагою у використанні байт-коду є портативність. Однак додаткова вартість інтерпретації означає, що інтерпретовані програми майже завжди працюватимуть повільніше, ніж ті, що складені в машинний код, через що Java здобула репутацію “повільної” мови. Однак цей розрив значно скоротився після впровадження декількох методів оптимізації в тогочасних JVM реалізаціях.

Один з таких методів представляє собою своєчасну компіляцію (JIT), яка перетворює байт-код Java в машинний код під час першого запуску програми, а потім кешування її. Складні віртуальні машини також використовують динамічну рекомпіляцію, в якій віртуальна машина аналізує поведінку запущеної програми та вибірково перекомпілює та оптимізує певні її частини. Динамічна рекомпіляція може досягти більш високого рівня оптимізації, ніж статична компіляція, оскільки динамічний компілятор може робити оптимізацію на основі знань про середовище виконання та завантажені класи. Крім того, він може виявити так звані гарячі точки - частини програми, часто внутрішні цикли, на виконання яких потрібен найбільше часу. Компіляція JIT та динамічна рекомпіляція збільшують швидкість Java-додатків, не втрачаючи портативності.

На відміну від C ++, Java більш орієнтована на об'єкти. Усі дані та дії згруповані в об’єктні класи. Винятки з повної об'єктивності (як у Smalltalk) - це примітивні типи (int, float тощо). Це було свідоме рішення мовних дизайнерів збільшити швидкість. Через це Java не вважається повністю об'єктно-орієнтованою мовою.

У Java всі об'єкти походять від основного об'єкта (він називається просто Об'єкт), від якого вони успадковують основну поведінку та властивості.

Хоча множинне успадкування вперше доступне в C ++, у Java можливе лише одне успадкування, що виключає можливість конфліктів між членами класу (методами та змінними), які успадковуються від базових класів.

Проектувальники задумали мовою Java замінити C ++, спадкоємцем об'єкта C. Вони почали з аналізу властивостей C ++, які викликають більшість помилок для створення простої, безпечної та безпроблемної мови програмування.

У Java існує система винятків або ситуацій, коли програма стикається з несподіваними труднощами, наприклад:

- операції над елементом масиву за його межами або над порожнім елементом;
- зчитування з важкодоступного каталогу чи неправильної URL-адреси;
- ввід користувачем недопустимих даних.

Однією з особливостей концепції віртуальної машини є те, що помилки (винятки) не призводять до повного колапсу системи. Крім того, є інструменти, які “приєднуються” до середовища виконання і кожного разу, коли виникає певний виняток, записують інформацію з пам'яті для налагодження програми. Ці автоматизовані засоби обробки винятків надають основну інформацію про виключення в додатках Java.

Однак мова програмування Java не рекомендується використовувати в системах, які можуть спричинити смерть, травми або серйозні фізичні травми (наприклад, програмне забезпечення для управління атомними електростанціями, польотами, системами життєзабезпечення) через ненадійність програм написано мовою програмування.

Java використовує автоматичний збирач сміття (GC) для управління пам'яттю протягом життєвого циклу об'єкта. Програміст вирішує, коли створити об'єкти, а віртуальна машина відповідає за звільнення пам'яті після того, як об'єкт стає непотрібним. Якщо на конкретний об’єкт не залишилося

жодних посилань, сміттєзбірник може автоматично видалити його з пам'яті. Однак витік пам'яті все ще може відбуватися, якщо код, написаний програмістом, має посилання на об'єкти, які більше не потрібні, наприклад, на об'єкти, що зберігаються в існуючих контейнерах.

Збір сміття дозволений у будь-який час. В ідеалі це відбувається під час бездіяльності програми. Збір сміття автоматично вимушений, коли в купі бракує вільної пам’яті для розміщення нового об’єкта, що може призвести до замерзання кількох секунд. Тому існують реалізовані віртуальні машини Java зі збирачем сміття, спеціально розроблені для програмування систем реального часу.

Java не підтримує покажчики стилів C / C ++. Це робиться для безпеки та надійності, щоб сміттєзбірник міг переміщувати об’єкти вказівника.

Android - це операційна система та платформа для мобільних телефонів та планшетів, створена Google на основі ядра Linux. За підтримки Альянсу відкритих слухавок (OHA).

Хоча Android базується на ядрі Linux, він відрізняється від спільноти Linux та інфраструктури Linux. Основним елементом цієї операційної системи є реалізація віртуальної машини Dalvik Java, і все програмне забезпечення та додатки базуються на цій реалізації Java.

Операційна система Android була встановлена у 84% смартфонів, проданих у кінці 3 кварталу в 2014 році.

В 2017 році операційна система Android стала найпопулярнішою операційною системою, з якої запустили Інтернет. Так, 37,91% користувачів прийшли в Інтернет з Windows, а з Android - 37,93% користувачів. На азійському ринку показники вищі - 29,2% та 52,2% відповідно.

Android, Inc. була заснована в Пало-Альто, штат Каліфорнія, в жовтні 2003 року Енді Рубін (співзасновник компанії Danger), Річард Майнер (співзасновник Wildfire Communications, Inc.), Нік Сірс (колишній віце-президент T-Mobile) та Кріс Уайт (керівник розробки та розробки інтерфейсу

на WebTV) розробити, за словами Рубіна, “розумніші мобільні пристрої, які краще знають місце розташування власника та його уподобання”. Ранніми намірами компанії були розробити вдосконалену операційну систему для цифрових камер, але було зрозуміло, що ринок пристроїв недостатньо великий, і вони зосередили свої зусилля на розробці операційних систем для мобільний пристроїв, аби бути конкурентом Windows Mobile та Symbian (тоді iPhone ще не був випущений компанією Apple). Недивлячись на минулі здобутки засновників та ранніх співробітників, Android, Inc. працювала таємно, показуючи лише, що вона працює над програмним забезпеченням для мобільних телефонів. Того ж року Рубіну закінчилися гроші. Стів Перлман, близький друг Рубіна, приніс йому в конверті 10 000 доларів готівкою і від своєї долі в компанії відмовився.

У 2005 році Google придбав Android, Inc. Усі засновники цієї стартап-компанії перейшли працювати в Google. У той час про Android, Inc. було мало відомо окрім розробки програмного забезпечення для мобільних телефонів. Така подія призвела до чуток, що Google планує вийти на ринок мобільних телефонів, але було незрозуміло, що компанія планує там робити.

У Google група під керівництвом Рубіна розробила операційну систему Android на базі операційної системи Linux (ядра v2.6). Дану ОС вони рекомендували мобільним операторам та розробникам телефонів як розширювану та гнучку систему. На той час Google планував співпрацювати з низкою розробників апаратного та програмного забезпечення та відкритий для співпраці з мобільними операторами.

У грудні 2006 року знову поширилися чутки, що Google перейде на ринок мобільних телефонів. У звітах BBC і The Wall Street Journal вказується, що Google хоче розмістити пошук і програмне забезпечення Google на мобільних телефонах, і компанія постійно наполегливо працює над досягненням цієї мети.

Пізніше в пресі та засобах масової інформації на просторах інтернету почали ходити чутки, про те що компанія Google займається розробкою

телефонів під власною маркою. За ними слідували інші, які заявляли, що Google визначив технічні характеристики та вже представляв прототипи розробникам телефонів та мобільним операторам. Повідомлялося, що буде реалізовано близько 30 прототипів. Network World повідомляє, що телефон Google справді є відкритим операційним телефоном, на відміну від подібних продуктів, таких як iPhone. Проект по створенню смартфона з використанням відкритого коду, включаючи використання ядра Linux.

5 листопада 2007 року Альянс відкритих слухавок (OHA) оголосив про намір розробити відкриті стандарти для мобільних пристроїв. Того ж дня концерн представив у якості свого першого продукту платформу для мобільних телефонів на базі Linux - Android.

У 2010 році Google запустив серію планшетів і смартфонів на базі ОС Android – серію Nexus, розробкою якої займається партнер компанії. Першою компанією, яка вирішила випустити перший смартфон у лінійці Nexus - Nexus One, була HTC. Наступні пристрої серії Nexus, Nexus S та Galaxy Nexus, були випущені Samsung наприкінці 2010 та 2011 років відповідно. У 2012 році серію було оновлено новими пристроями: смартфоном Nexus 4 та планшетом Nexus 10 виробництва LG та Samsung відповідно. У серії Nexus компанія Google втілює свої флагманські пристрої Android, демонструючи останні версії програмного забезпечення та апаратних функцій Android.

13 березня 2013 року Ларрі Пейдж оголосив у своєму блозі, що Енді Рубін покидає підрозділ Android, щоб взяти на себе нові проекти в Google. Його замінив Сундар Пічай, який також продовжує бути керівником Google Chrome, який розробляє ОС Chrome.

Nexus 7 другого покоління - перший планшет з операційною системою Android 4.3. Однією з нововведень, реалізованих для цієї версії, була можливість створювати декілька користувацьких профілів, кожен з яких мав різні права доступу, наприклад, для запобігання доступу дітей до небажаного вмісту. Крім того, Android 4.3 підтримує стандарт Bluetooth Smart, який

спрямований на розширення традиційної сфери використання бездротового інтерфейсу Bluetooth.

12 листопада 2007 року OHA представила комплект програм для розробки програмного забезпечення для Android, який включав інструменти для розробки програмного забезпечення та налагодження, бібліотеки, емулятор, документацію, зразкові програми, навчальний посібник, поширені запитання та інше. Для розробки вам потрібно завантажити платформу Android SDK для платформ x86 з Windows XP або Vista, Mac OS X 10.4.8 або новішої версії, або Ubuntu Linux (Dapper Drake або вище). Він також повинен працювати в інших дистрибутивах Linux, але безпосередньо ця функція не підтримується. У вас також повинен бути встановлений Eclipse 3.2 або новішої версії, з Java Development Tools та плагіном Android SDK або Java, Python 2.2, Apache Ant, Javac версії 1.5 та 1.6.

21 жовтня 2008 року альянс OHA випустив вихідний код Android. Випуск включає весь стек Android: операційну систему, проміжне програмне забезпечення та основні кінцеві програми, написані на Java. Загальна кількість необробленого коду Android становила 2,1 ГБ.

“Найкраща ліцензія” для операційного коду Android - ліцензія Apache 2.0.

У грудні 2019 року компанія Google запустила сервіс cs.android.com, призначений для пошуку необробленого коду у сховищах git, пов’язаних із платформою Android. Пошук враховує різні класи елементів, знайдені в коді, і результат відображається візуально з підсвічуванням синтаксису, можливістю переміщення між посиланнями та переглядом історії змін. Наприклад, ви можете натиснути на ім’я функції у вашому коді вашого проекту та перейти до місця визначення цієї функції або переглянути, де ще вона викликається. Ви також можете перемикатися між різними гілками та оцінювати зміни між ними.

У травні 2010 року було досягнуто обмеження в 100 тис. Активацій на день. У грудні 2010 року в день було активовано 300 000 пристроїв. У травні на конференції Google I / O було оголошено статистику, згідно з якою близько 400

тис. Нових пристроїв на базі платформи Android щодня активуються. У липні 2011 року віце-президент Google, відповідальний за розробку платформи Android, Енді Рубін оголосив про новий ліміт в 500 000 активацій на день, платформа на 4,4% щотижня. Статистика активації включає лише інформацію про першу реєстрацію нових пристроїв, які постачаються з набором послуг Google. Пристрої з прошивкою без набору додатків Google, що виробляються деякими азіатськими виробниками, не включаються до статистики.

Загалом станом на липень 2011 року в мережах 215 операторів зв'язку було продано та розповсюджено понад 100 мільйонів пристроїв Android, проданих 36 виробниками. Загальна кількість моделей пристроїв на базі платформи Android досягла 310. Всього було продано понад 200 мільйонів пристроїв Android. Каталог Google Play перевершив позначку в 200 тисяч додатків. Загалом із Google Play встановлено близько 4,5 мільярдів примірників програм.

За даними Google, у вересні 2015 року було активовано 1,4 мільярда телефонів Android.

22 жовтня 2008 року Google оголосив про відкриття Android Market - інтернет-магазину Android-додатків; розробники будуть отримувати 70% від прибутку, а стільникові оператори - 30%. Станом на березень 2016 року Google Play містила понад 2 мільйони програм. Google класифікує користувачів системи Andorid Market на 4 категорії:

- розробники безплатних програмних продуктів;
- розробники платних програмних продуктів;
- користувачі безплатних програмних продуктів;
- користувачі платних програмних продуктів;

Для кожної з цих груп на Android Market була окрема ліцензійна угода.

У березні 2012 року Google перейменував Android Market на Google Play.

Google Play також дозволяє розробникам робити додатки недоступними для деяких пристроїв.

Продажі перших смартфонів Android в Україні розпочалися 15 січня 2009 р. Першим офіційним смартфоном Android в Україні вважається HTC Hero, хоча в Україні є різні дати його появи: 28 вересня 2009 р., 19 жовтня 2009 р. Листопад 2009 року. (водночас, офіційною датою виходу на український ринок корпорація HTC оголосила 25 травня 2010 року, коли були представлені Android-смартфони HTC Legend, HTC Desire, HTC Wildfire). HTC Hero також присутній на українському ринку як операторське рішення принаймні з 1 березня 2010 року. Другим офіційним смартфоном є Samsung Galaxy Spica, який був представлений 19 листопада 2009 року та надійшов у продаж у грудні. Оператор Huawei IDEOS C8150 став першим офіційним смартфоном на базі ОС Android з підтримкою CDMA / EV-DO технологій.

Поява Google Play на офіційних пристроях, що імпортуються в Україну (з початку виробництва Samsung, потім HTC), компанія “Android Україна” передбачила кінець січня 2010 року. 13 січня 2010 року Samsung оголосила про початок поставок в Україну смартфона з можливістю українцям мати доступ до Android Market - Galaxy Spica i5700. 8 грудня 2010 року розповсюдилась інформація про те, що платну версію Android Market для України проходять тестування.

12 травня 2011 року можливість придбати додатки в Google Play стала доступною для жителів України. Усі ціни платних заявок для українців вказані в гривнях. Для здійснення платежу необхідна платіжна картка, зареєстрована в Google Checkout.

Починаючи з версії 1.6, Android має повноцінну офіційну українську локалізацію. Якщо у вас немає можливості вибрати українську мову на своєму смартфоні, ви можете легко додати її, завантаживши програму під назвою MoreLocale2 з магазину Google Play Apps.

Для розробки Android-додатку використовувалось інтегроване середовище розробки Android Studio, яке надає широкий спектр можливостей

для розробки мобільних додатків для пристроїв з операційною системою Android на двох мовах програмування: Java та Kotlin.

Android Studio є інтегрованим середовищем розробки (IDE) для Android пдатформи, яке було представлене на конференції Google I / O 16 травня 2013 року керівником продуктів Google Еллі Пауерс. 8 грудня 2014 року Google випустила перший стабільний випуск Android Studio 1.0.

Android Studio замінив плагін ADT для платформи Eclipse. Навколишнє середовище базується на вихідному коді продукту IntelliJ IDEA Community Edition, розробленого JetBrains. Android Studio розроблений як частина відкритої моделі розробки та поширюється за ліцензією Apache 2.0.

Бінарні збірки підготовлені для Linux (для тестування використовується Ubuntu), Mac OS X та Windows. Інтегроване середовище надає інструменти для розробки програмного забезпечення для смартфонів, планшетів, телевізорів (Android TV), окулярів Google Glass, Android Wear пристроїв та інфо-розважальних систем для автомобілів (Android Auto). Для програм, спочатку розроблених за допомогою Eclipse та ADT Plugin, було підготовлено інструменти які дозволяли автоматично імпортувати існуючі проекти в Android Studio.

Дане IDE пристосоване для виконання типових завдань, які можуть бути вирішеним під час розробки додатків для Android платформи. Сюди входять інструменти для проектування додатків та інструменти для спрощення тестування програмного забезпечення на сумісність з іншими версіями даної платформи, які працюють на пристроях із екранами різної роздільної здатності (планшети, смартфони, ноутбуки, годинник, окуляри тощо). На додаток до функцій, наявних в IntelliJ IDEA, Android Studio має ряд додаткових функцій, наприклад, нову єдину підсистему для складання, тестування та розгортання програм на основі інструментів збирання Gradle та підтримки використання засобів безперервної інтеграції.

Для прискорення розробки додатків існує колекція типових елементів інтерфейсу та візуальний редактор для їх компонування, який забезпечує зручний перегляд різних станів інтерфейсу програми (наприклад, ви можете побачити, як буде виглядати інтерфейс для різних версій Android та різні розміри екрана). Щоб створити нестандартні інтерфейси, є майстер створення власних елементів дизайну, які підтримують використання шаблонів. У середовищі є вбудовані функції для завантаження типових прикладів коду з GitHub.

Він також включає в себе вдосконалені інструменти рефакторингу, перевірку сумісності з минулими випусками, проблеми з продуктивністю, моніторинг використання пам'яті та оцінку зручності використання, з урахуванням платформи Android. У редактор додано швидкий режим редагування. Система виділення, статичного аналізу та виявлення помилок розширена для підтримки API Android. Вбудована підтримка оптимізатора коду ProGuard. Вбудовані засоби генерації цифрового підпису. Надається інтерфейс для управління перекладами на інші мови.

Пізніше деякі функції будуть розгорнуті для користувачів у міру розвитку програмного забезпечення; В даний час надаються наступні функції:

- живі макети: редактор WYSIWYG - кодування в реальному часі - візуалізація програми в режимі реального часу;
- консоль розробника: поради щодо оптимізації, допомога в перекладі, відстеження, кампанії та рекламні кампанії - показники Google Analytics;
- покрокові релізи та резерви бета релізів;
- базування на збиральнику проектів Gradle;
- швидкі виправлення та android-орієнтований рефакторинг;
- утиліти для зручності використання, сумісності версій, охоплення продуктивності та інших проблем;
- аикористання підписів до програм та можливостей ProGuard;

- шаблони для створення компонентів та Android дизайнів;
- багатий редактор макетів, що дозволяє користувачам перетягувати компоненти інтерфейсу користувача, як опцію, одночасно переглядати макети в різних конфігураціях екрана.

Перш за все було створено макети для кожного Activity (активного вікна програми).

Макет визначає візуальну структуру інтерфейсу користувача, наприклад, інтерфейс користувача операції або віджет програми. Є два способи оголошення макета:

- оголошення елементів інтерфейсу користувача в XML. Android має зручний посібник з елементами XML для класів View та їх підкласів, таких як віджети та макети;
- інстанціювати елементи під час виконання. Ваша програма може програмно створювати об’єкти View та ViewGroup (а також керувати їх властивостями).

Платформа Android забезпечує гнучкість у використанні будь-якого з цих методів для оголошення інтерфейсу користувача програми та управління ним. Наприклад, ви можете оголосити макети за замовчуванням у XML, включаючи елементи екрана, які відображатимуться у макетах та їх властивості. Потім ви можете додати код до програми, яка дозволяє змінювати стан об’єктів на екрані (включаючи ті, які оголошені в XML) під час виконання.

Всього додаток містить три Activity: MainActivity, AuthorizationFirstStepActivity, AuthorizationSecondStepActivity – усі ці файли мають розширення java, а їх макети – xml.

MainActivity – головне активне вікно. При увімкненні застосунку воно відкривається першим. Для налаштування головного вікна потрібно редагувати manifest файл конфігурацій. Макет даного Activity містить лейбл – текстове поле для коротких назв чого-небудь, у даному проекті вказує на назву групи чи викладача, текстове поле куди вводиться текст, сказаний користувачем, велике

текстове поле, куди асистент виводить результат запиту користувача, іконка мікрофону, клікнувши на яку відкриється меню для запису голосу від Google (рис. 3.1).

Рисунок 3.1 – Інтерфейс MainActivity

Інтерфейс активних вікон програми розробляється з допомогою XML, на рис. 3.2 зображений xml код для MainActivity.

activity_main.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@drawable/bg"
    android:paddingLeft="20dp"
    android:paddingTop="1dp"
    android:paddingRight="20dp"
    android:paddingBottom="20dp"
    tools:context=".MainActivity">

    <TextView
        android:id="@+id/mainActivityTextVoiceInput"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_below="@+id/mainActivityLabel"
        android:layout_marginTop="10dp"
        android:hint="@string/press_and_speak"
        android:textColor="#fff"
        android:textColorHint="#b0b0b0"
        android:textSize="40sp" />

    <TextView
        android:id="@+id/mainActivityOut"
        android:layout_width="377dp"
        android:layout_height="341dp"
        android:layout_below="@id/mainActivityTextVoiceInput"
        android:layout_marginTop="40dp"
        android:textColor="#fff"
        android:textSize="40sp" />
```

Рисунок 3.2 – XML код макету MainActivity, аркуш 1

```xml
    <ImageButton
        android:id="@+id/mainActivityVoiceBtn"
        android:layout_width="120sp"
        android:layout_height="120sp"
        android:layout_alignParentEnd="true"
        android:layout_alignParentRight="true"
        android:layout_alignParentBottom="true"
        android:layout_marginEnd="14dp"
        android:layout_marginRight="14dp"
        android:layout_marginBottom="21dp"
        android:background="@null"
        android:contentDescription="@string/todo"
        android:scaleType="fitCenter"
        android:src="@drawable/ic_mic" />

    <TextView
        android:id="@+id/mainActivityLabel"
        android:layout_width="367dp"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_marginTop="1dp"
        android:gravity="end"
        android:textColor="#fff"
        android:textSize="30sp"
        android:textStyle="bold|italic" />

</RelativeLayout>
```

Рисунок 3.2 – XML код макету MainActivity, аркуш 2

AuthorizationFirstStepActivity – активне вікно першого кроку реєстрації. Містить заголовок та дві кнопки (“Студент”, “Викладач”), кожна з яких записує

екстра дані в клас Intent і переходить на активнее вікно для другого кроку реєстрації (рис. 3.3).

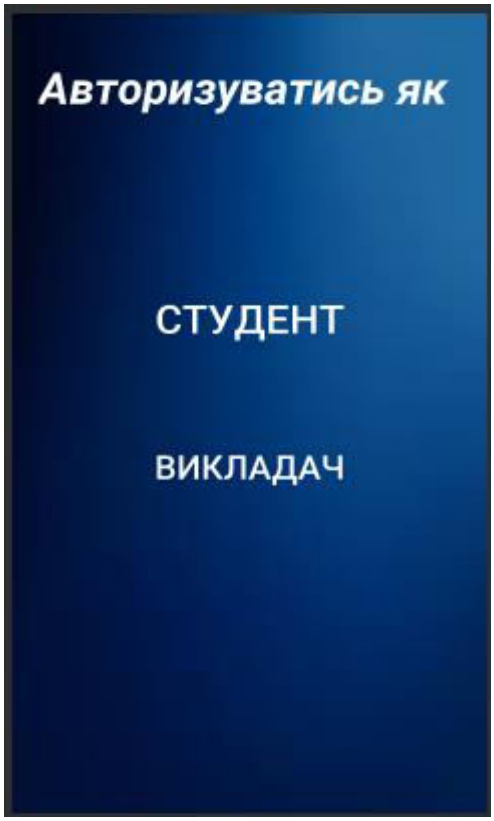

Рисунок 3.3 – Інтерфейс AuthorizationFirstStepActivity

XML код макету AuthorizationFirstStepActivity доволі легкий у розумінні та низький у плані ресурсозатрат при обробці пристроєм (рис. 3.4).

activity_autorization_first_step.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@drawable/bg"
    tools:context=".AuthorizationFirstStepActivity">

    <TextView
        android:id="@+id/authorizationFirstStepActivityTitle"
        style="@style/Widget.AppCompat.TextView"
        android:layout_width="364dp"
        android:layout_height="96dp"
        android:text="@string/sign_in_as"
        android:textColor="#fff"
        android:textColorLink="#fff"
        android:textSize="40sp"
        android:textStyle="bold|italic"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.493"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.063" />

    <Button
        android:id="@+id/authorizationFirstStepActivityBtnStudent"
        style="@style/Widget.AppCompat.Button.Borderless"
        android:layout_width="299dp"
        android:layout_height="136dp"
        android:text="@string/student"
        android:textColor="#fff"
        android:textSize="36sp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/authorizationFirstStepActivityTitle"
        app:layout_constraintVertical_bias="0.162" />


    <Button
        android:id="@+id/authorizationFirstStepActivityBtnTeacher"
        style="@style/Widget.AppCompat.Button.Borderless"
        android:layout_width="299dp"
        android:layout_height="136dp"
        android:background="?attr/colorControlNormal"
        android:text="@string/teacher"
        android:textColor="#fff"
        android:textSize="30sp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/authorizationFirstStepActivityBtnStudent"
        app:layout_constraintVertical_bias="0.0" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

Рисунок 3.4 – XML код макету AuthorizationFirstStepActivity

AuthorizationSecondStepActivity – активне вікно другого кроку реєстрації. Інтерфейс доволі простий, але зі складним функціоналом: містить заголовок, випадаючий список, елементи якого генеруються взалежності від першого кроку реєстрації, та кнопка “Підтвердити” (рис. 3.5).

Рисунок 3.5 – Інтерфейс AuthorizationSecondStepActivity

Як і у випадку макету першого кроку реєстрації, XML код макету другого кроку доволі простий у розумінні та оптимізований (рис. 3.6).

activity_authorization_second_step.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@drawable/bg"
    tools:context=".AuthorizationSecondStepActivity">

    <Spinner
        android:id="@+id/authorizationSecondStepActivitySelect"
        style="@style/Widget.AppCompat.Light.Spinner.DropDown.ActionBar"
        android:layout_width="361dp"
        android:layout_height="51dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.26" />

    <TextView
        android:id="@+id/authorizationSecondStepActivityTitle"
        style="@style/Widget.AppCompat.TextView"
        android:layout_width="345dp"
        android:layout_height="72dp"
        android:layout_marginBottom="282dp"
        android:gravity="center"
        android:text="@string/choose_from_list"
        android:textColor="#fff"
        android:textSize="36sp"
        android:textStyle="bold|italic"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.129" />
```

Рисунок 3.6 – XML код макету AuthorizationSecondStepActivity, аркуш 1

```
    <Button
        android:id="@+id/authorizationSecondStepActivityBtnApply"
        style="@style/Widget.AppCompat.Button.Borderless"
        android:layout_width="300dp"
        android:layout_height="142dp"
        android:layout_marginTop="188dp"
        android:text="@string/apply"
        android:textColor="#fff"
        android:textSize="36sp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/authorizationSecondStepActivitySelect" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

Рисунок 3.6 – XML код макету AuthorizationSecondStepActivity, аркуш 2

Функціонал додатку доволі обширний. Він ділиться на шість файлів з розширенням java. Функціонал MainActivity як і будь який функціонал потрібно прописувати в методі onCreate().

Тут відбувається ініціалізація змінних, тобто зчитуються всі поля з макету і їх об'єкти записуються в змінні, потім запис в зміну localData записується JSON об'єкт, отриманий з файлу data.json з допомогою методу getJsonFromFile об’єкта класу JsonOps (даний клас створений власноруч), потім відбувається перевірка, чи є у localData (оскільки це вже JSON об’єкт) ключ “error”, який створюється при відсутності файлу вказаному як параметр функції getJsonFromFile, якщо є, то з допомогою об’єкту класу Intent та функції startActivity (стандартна функція Android Studio) переходимо на вікно першого кроку реєстрації, якщо ключ “error” відсутній, то вішаємо на іконку з мікрофоном OnClickListener (по суті об’єкт класу View.OnClickListener який відслідковує подію кліку), який при кліку на неї буде виконувати функцію speak() (рис. 3.7).

```
public class MainActivity extends AppCompatActivity {
    private static final int REQUEST_CODE_SPEECH_INPUT = 1000;
    TextView out, label, voiceInput;
    ImageButton voiceBtn;
    String baseUrl = "https://82dc25c7.ngrok.io/";

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        out = findViewById(R.id.mainActivityOut);
        label = findViewById(R.id.mainActivityLabel);
        voiceInput = findViewById(R.id.mainActivityTextVoiceInput);
        voiceBtn = findViewById(R.id.mainActivityVoiceBtn);

        JsonOps jsonOps = new JsonOps();

        try {
            JSONObject localData = jsonOps.getJsonFromFile(this, "data.json");
            if(localData.has("error")){
                startActivity(new Intent(this, AuthorizationFirstStepActivity.class));
            } else {
                label.setText(localData.getString("label"));
                voiceBtn.setOnClickListener(new View.OnClickListener() {
                    @Override
                    public void onClick(View v) {
                        speak();
                    }
                });
            }
        } catch (JSONException e) {
            e.printStackTrace();
        }
    }
```

Рисунок 3.7 – Перезаписування методу onCreate

Метод speak() полягає у створенні діалогового вікна Google, який буде зчитувати голосові команди користувача, з допомогою класу Intent, потім з допомогою функції startActivityForResult() виконуватиме дану дію і результат обробки даної дії можна буде отримати та обробити з допомогою перезаписуванням методу onActivityResult, де буде відбуватись зчитування даних з результату і виконання методу assistant() куди буде в якості параметру відсилатись отриманий результат (рис. 3.8).

```
public void speak(){
    Intent intent = new Intent(RecognizerIntent.ACTION_RECOGNIZE_SPEECH);
    intent.putExtra(RecognizerIntent.EXTRA_LANGUAGE_MODEL, RecognizerIntent.LANGUAGE_MODEL_FREE_FORM);
    intent.putExtra(RecognizerIntent.EXTRA_LANGUAGE, Locale.getDefault());
    intent.putExtra(RecognizerIntent.EXTRA_PROMPT, "Привіт, чим я можу Вам допомогти?");

    try {
        startActivityForResult(intent, REQUEST_CODE_SPEECH_INPUT);
    } catch (Exception e){
        Toast.makeText(this, ""+e.getMessage(), Toast.LENGTH_SHORT).show();
    }
}

@Override
protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {
    super.onActivityResult(requestCode, resultCode, data);

    if (requestCode == REQUEST_CODE_SPEECH_INPUT) {
        if (resultCode == RESULT_OK && null != data) {
            ArrayList<String> result = data.getStringArrayListExtra(RecognizerIntent.EXTRA_RESULTS);
            assert result != null;
            try {
                assistant(result.get(0));
            } catch (JSONException e) {
                e.printStackTrace();
            }
        }
    }
}
```

Рисунок 3.8 – метод speak()

Метод assistant() переводить отриманий текст у нижній регістр, виводить його в поле вводу тексту, отримує статус користувача і взалежності від нього виконує функцію studentAssistant() або teacherAssistant(), куди в якості параметру передає текст в нижньому регістрі, в разі помилки з допомогою класу Toast відсилає сповіщення про помилку (рис. 3.9).

```
@SuppressLint("SetTextI18n")
public void assistant(String command) throws JSONException {
    String text = command.toLowerCase();
    voiceInput.setText(text);

    String status = new JsonOps().getJsonFromFile(this, "data.json")
            .getString("status");

    switch (status){
        case "student":
            studentAssistant(text);
            break;
        case "teacher":
            teacherAssistant(text);
            break;
        default:
            Toast.makeText(this, "Some error!", Toast.LENGTH_LONG).show();
    }
}
```

Рисунок 3.9 – метод assistant()

Методи studentAssistant() та teacherAssistant() ідентичні, відрізняються лиш URL адресами для відправлень HTTP запитів на back-end частину проекту, тобто на API. Методи отримують час відправлення запиту, генерують URL адресу, з допомогою методу insertJsonToFileFromUrl() відправляють запит, записують результат у файл “api.json”, отримують JSON об’єкт з цього файлу, перевіряють його на наявність помилок, при їх присутності на центральну частину екрану у велике текстове поле виводять повідомлення про помилку, при відсутності помилок – виводять отриманий результат (рис. 3.10).

```
public void studentAssistant(String text) throws JSONException {
    String group = label.toString();
    SimpleDateFormat sdf = new SimpleDateFormat("HH:mm", Locale.getDefault());
    String currentTime = sdf.format(new Date());
    JsonOps jsonOps = new JsonOps();

    String url = baseUrl + "api/";
    url += text.contains("де") && text.contains("вільн") && text.contains("ауд") ?
            "where_free_auditorium/time=" + currentTime :
            text.contains("де") && text.contains("пара") ?
                    "where_subject/group="+group+"&time=" + currentTime : null;

    jsonOps.insertJsonToFileFromUrl(this, url);
    JSONObject jsonObject = jsonOps.getJsonFromFile(this, "api.json");

    String outText = jsonObject.has("error") ? "Не знайдено результатів" :
            jsonObject.get("answer").toString();
    out.setText(outText);
}

public void teacherAssistant(String text) throws JSONException {
    SimpleDateFormat sdf = new SimpleDateFormat("HH:mm", Locale.getDefault());
    String currentTime = sdf.format(new Date());
    JsonOps jsonOps = new JsonOps();

    String url = baseUrl + "api/";
    url += text.contains("де") && text.contains("вільн") && text.contains("ауд") ?
            "where_free_auditorium/time=" + currentTime : null;

    jsonOps.insertJsonToFileFromUrl(this, url);
    JSONObject jsonObject = jsonOps.getJsonFromFile(this, "api.json");

    String outText = jsonObject.has("error") ? "Не знайдено результатів" :
            jsonObject.get("answer").toString();
    out.setText(outText);
}
```

Рисунок 3.10 – методи studentAssistant() і teacherAssistant()

AuthorizationFirstStepActivity має доволі простий і зрозумілий алгоритм: зчитує об’єкти кнопок “Студент” та “Викладач”, вішає на них OnClickListener, який при кліку буде виконувати функцію nextStep() і передавати туди параметр у вигляді строки “student” або “teacher”, суть функції полягає у записі екстра

даних в клас Intent і перехід на наступне активне вікно, в даному випадку активне вікно другого кроку реєстрації (рис. 3.11).

```
public class AuthorizationFirstStepActivity extends AppCompatActivity {

    Button studentBtn, teacherBtn;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_autorization_first_step);

        studentBtn = findViewById(R.id.authorizationFirstStepActivityBtnStudent);
        teacherBtn = findViewById(R.id.authorizationFirstStepActivityBtnTeacher);

        studentBtn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                nextStep("group");
            }
        });

        teacherBtn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                nextStep("teacher");
            }
        });
    }

    public void nextStep(String status){
        Intent intent = new Intent(this, AuthorizationSecondStepActivity.class);
        intent.putExtra("status", status);
        startActivity(intent);
    }
}
```

Рисунок 3.11 – Код AuthorizationFirstStepActivity

AuthorizationSecondStepActivity отримує об’єкт випадаючого списку, кнопки підтвердження, екстра дані класу Intent (статус користувача), відповідно від отриманих даних генерує URL адресу для HTTP запиту на API проекту, отримує списки викладачів або студентів, заповнює випадаюче меню елементами зі списку, отримує вибране користувачем значення з випадаючого списку і записує отримані дані в файл “data.json” методом insertJsonToFile класу JsonOps (рис. 3.12).

```java
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_authorization_second_step);

    applyBtn = findViewById(R.id.authorizationSecondStepActivityBtnApply);
    selectInput = findViewById(R.id.authorizationSecondStepActivitySelect);
    Intent intent = getIntent();
    String status = intent.getStringExtra("status");

    JsonOps jsonOps = new JsonOps();
    String baseUrl = "https://82dc25c7.ngrok.io";

    jsonOps.insertJsonToFileFromUrl(this,baseUrl+"/api/get_"+status+"s");

    try {
        JSONArray jsonArray = jsonOps.getJsonFromFile(this, "api.json")
                .getJSONArray("answer");
        List<String> selectData =  new ArrayList<String>();
        for (int i=0; i < jsonArray.length(); i++){
            selectData.add(jsonArray.getString(i));
        }
        ArrayAdapter<String> adapter = new ArrayAdapter<String>(
                this, android.R.layout.simple_spinner_item, selectData);
        adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
        selectInput.setAdapter(adapter);

        JSONObject data = new JSONObject();
        data.put("status", "student");
        data.put("label", selectInput.getSelectedItem().toString());

        jsonOps.insertJsonToFile(this, "data.json", data);

    } catch (JSONException e) {
        e.printStackTrace();
    }

    applyBtn.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            nextStep();
        }
    });
}
```

Рисунок 3.12 – код AuthorizationSecondStepActivity

Для роботи з локальними JSON файлами використовуються три методи класу JsonOps: insertJsonToFile, getJsonFromFile, insertJsonToFileFromUrl.

Метод insertJsonToFile приймає шлях до файлу та JSON об’єкт який повинен бути записаним, якщо файлу не існує, то створюється новий, якщо існує – то перезаписується (рис. 3.13).

```
public void insertJsonToFile(Context context, String filename, JSONObject jsonObject ){
    File file = new File(context.getFilesDir(), filename);
    try {
        if(!file.exists()){
            file.createNewFile();
        }
        FileWriter fileWriter = new FileWriter(file.getAbsoluteFile());
        BufferedWriter bufferedWriter = new BufferedWriter(fileWriter);
        bufferedWriter.write(jsonObject.toString());
        bufferedWriter.close();
    } catch (IOException e) {
        Toast.makeText(context, e.getMessage(), Toast.LENGTH_LONG).show();
    }
}
```

Рисунок 3.13 – метод insertJsonToFile

Метод getJsonFromFile отримує шлях до файлу, зчитує файл якщо він знайдений, якщо ні – повертає JSON об’єкт з ключем “error” (рис. 3.14).

```
public JSONObject getJsonFromFile(Context context, String filename) throws JSONException {
    File file = new File(context.getFilesDir(), filename);
    StringBuilder stringBuilder = new StringBuilder();
    try {
        FileReader fileReader = new FileReader(file.getAbsoluteFile());
        BufferedReader bufferedReader = new BufferedReader(fileReader);
        String line = "";
        while ((line = bufferedReader.readLine()) != null) {
            stringBuilder.append(line).append("\n");
        }
        bufferedReader.close();

    } catch (FileNotFoundException e) {
        Toast.makeText(context, e.getMessage(), Toast.LENGTH_LONG).show();
        return new JSONObject("{error: \"file not found\"}");
    } catch (IOException e) {
        Toast.makeText(context, e.getMessage(), Toast.LENGTH_LONG).show();
    }
    return new JSONObject(stringBuilder.toString());
}
```

Рисунок 3.14 – метод getJsonFromFile

Метод insertJsonToFileFromUrl приймає URL адресу, відправляє туди запит, отриману відповідь записує у файл “api.json” методом insertJsonToFile, при помилках записує JSON об’єкт з ключем “error” (рис. 3.15).

```
public void insertJsonToFileFromUrl(final Context context, String urlString) {
    RequestQueue requestQueue = Volley.newRequestQueue(context);
    JsonObjectRequest jsonObjectRequest = new JsonObjectRequest(
            Request.Method.GET,
            urlString, null,
            new Response.Listener<JSONObject>() {
                @Override
                public void onResponse(JSONObject response) {
                    insertJsonToFile(context, "api.json", response);
                }
            },
            new Response.ErrorListener() {
                @Override
                public void onErrorResponse(VolleyError e) {
                    Toast.makeText(context, e.getMessage(), Toast.LENGTH_LONG).show();
                    try {
                        insertJsonToFile(context, "api.json",
                                new JSONObject("{error: \"url not found\"}"));
                    } catch (JSONException ex) {
                        ex.printStackTrace();
                    }
                }
            });
    requestQueue.add(jsonObjectRequest);
}
```

Рисунок 3.15 – метод insertJsonToFileFromUrl

Для відправлення HTTP запитів використовується модуль Volley.

Volley – це HTTP бібліотека, яка робить мережу для додатків Android простішою і, головне, швидшою. Volley доступний на GitHub.

Volley має такі особливості:

- автоматично планує мережеві запити;
- містить декілька паралельних мережних зв’язків;
- має зі стандартною когерентністю кешу HTTP прозоре кешування відповідей пам'яті та диска;
- підтримує визначення пріоритетів запитів;
- дає можливість скасувати запит API. Ви можете скасувати один запит, а також можете встановити блоки або коло запитів для скасування;
- легкий у налаштуванні;
- містить сильне впорядкування, що спрощує правильне заповнення інтерфейсу користувача даними, отриманими асинхронно з мережі;
- має в собі інструменти для відстеження та налагодження.

Volley перевершує операції типу RPC, що використовуються щоб заповнити інтерфейс користувача, тобто отримати результати пошуку в якості структурованих даних. Він легко інтегрується з будь-яким протоколом і виходить з коробки з підтримкою необроблених рядків, зображень та JSON. Надаючи вбудовану підтримку основних функцій, Volley звільняє вас від написання коду і дозволяє зосередитись на логіці, характерній для вашої програми.

Volley не підходить для великих завантажувальних або потокових операцій, оскільки Воллі зберігає всі відповіді в пам'яті під час розбору. Для великих операцій із завантаженням продумайте альтернативу, наприклад, DownloadManager.

Основна бібліотека Volley розроблена на GitHub і містить основний конвеєр запитів, а також набір загальних утиліт, доступних на “панелі інструментів”.

3.2 Розробка back-end частини

Back-end частина проекту зроблена на мові програмування Python3 та фреймворці Flask, зокрема на модулі Flask RESTful.

Flask є мікро-фреймворком для веб-додатків, створених за допомогою Python. Він заснований на інструментах Werkzeug та механізмі шаблонів Jinja2. Поширюється за умовами BSD ліцензії.

В грудні 2016 року виходить стабільна версія Flask, яка має номер 0,12. Flask використовується для розробки проектів, таких як Pinterest, LinkedIn та сторінка спільноти Flask.

Flask називають мікро-фреймворком, оскільки він не потребує спеціальних інструментів чи бібліотек. Не вистачає рівня абстракції для роботи з базою даних, валідацією форми або іншими компонентами, які забезпечують широко використовувані функції через сторонні бібліотеки. Однак Flask має

підтримку розширень, які надають додаткові властивості так, ніби вони були доступні у Flask з початку його створення. Існує низка розширень для перевірки форм, підтримки різноманітних відкритих технологій аутентифікації, контролю процесу завантаження, встановлення об'єктно-реляційних зв’язків та декілька загальних інструментів для фреймворку. Розширення оновлюються частіше, ніж вихідний код.

Flask базується на Werkzeug WSGI, а також двигуні шаблонів Jinja2, який був створений як проект Pocoo відповідно в 2007 та 2008 роках, коли Ронакер та Георг Брандл створили систему дошки оголошень у Python.

Незважаючи на відсутність основного випуску, Flask серед шанувальників Python став надзвичайно популярним. В середині 2016 року це була найпопулярніша веб-рамка Python на GitHub.

Властивості мікро-фреймворку:

- містить відлагоджувач та сервер для розробки;
- містить вбудовану підтримку юніт-тестів;
- має в собі можливість управляти запитами RESTful;
- використовує шаблонізатор Jinja2;
- підтримує безпечні куки (сесії на клієнтській стороні);
- повністю відповідає WSGI 1.0;
- підтримує Unicode;
- має докладну документацію;
- сумісний з Google App Engine.

API проекту розбите на три допоміжні пакети, кожен з яких має по два модулі (рис. 3.16).

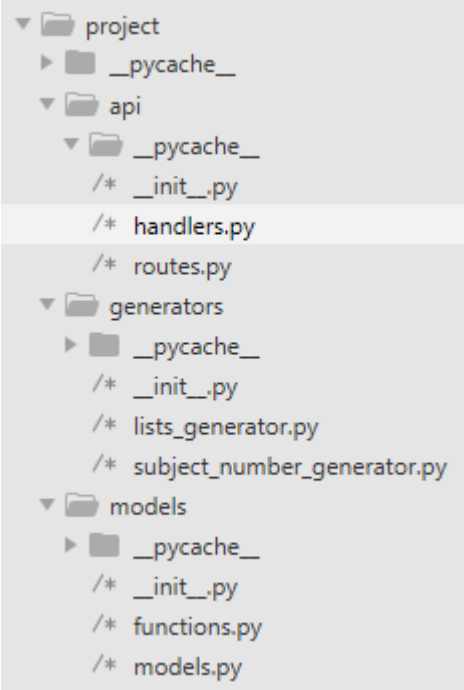

Рисунок 3.16 – структура API проекту

Папки з назвою “pyckache” та файли “init.py” не враховуються – це особливість мови програмування Python. Призначення кожного модуля описано вище (п. 2.3).

Модуль handler містить клас Handler та статичні методи які повертають результат роботи модуля functions (рис. 3.17).

```python
class Handler:
    @staticmethod
    def where_subject(group_name, current_time):
        result = get_subjects(current_time, group_name).first()
        return result[1].aud if result else None

    @staticmethod
    def where_free_auditorium(current_time):
        result = get_free_auds(current_time)
        return [res.aud for res in result] if result else None

    @staticmethod
    def what_teacher(group_name, current_time):
        result = get_subjects(current_time, group_name).first()
        return get_teacher_name_by_id(result[0].teacher_id) if result else None

    @staticmethod
    def what_subject(group_name, current_time):
        result = get_subjects(current_time, group_name).first()
        return result[0].name if result else None

    @staticmethod
    def is_teacher_here(teacher_name):
        result = get_subjects(teacher_name=teacher_name).all()
        return [{'aud': res[1].aud, 'num': res[1].subject_num} for res in result] if result else None

    @staticmethod
    def get_teachers():
        return get_teachers_names()

    @staticmethod
    def get_groups():
        return get_groups_names()
```

Рисунок 3.17 – модуль handler

Модуль routes використовує модуль flask RESTful і на основі нього створені класи які показують реакцію сервера на той чи інший HTTP запит, і в методі add_resource прописується клас та адреса яку клас повинен відстежувати (рис. 3.18).

```python
class CreateDatabase(Resource):
    @staticmethod
    def get():
        return {'answer': 'post password'}

    @staticmethod
    def post():
        json = request.get_json()
        if json['password'] != 'Деякий пароль':
            return {'answer': 'Method not allowed!'}, 405
        return {'answer': Handler().create_database()}


class UpdateDatabase(Resource):
    @staticmethod
    def get():
        return {'answer': 'post password'}

    @staticmethod
    def post():
        json = request.get_json()
        if json['password'] != 'Деякий пароль':
            return {'answer': 'Method not allowed!'}, 405
        return {'answer': Handler().update_database()}


class GetTeachers(Resource):
    @staticmethod
    def get():
        return {'answer': Handler().get_teachers()}


class GetGroups(Resource):
    @staticmethod
    def get():
        return {'answer': Handler().get_groups()}


api.add_resource(Index, '/')
api.add_resource(WhereSubject, '/api/where_subject/group=<string:group_name>&time=<string:current_time>')
api.add_resource(WhereFreeAuditorium, '/api/where_free_auditorium/time=<string:current_time>')
api.add_resource(WhatTeacher, '/api/what_teacher/group=<string:group_name>&time=<string:current_time>')
api.add_resource(WhatSubject, '/api/what_subject/group=<string:group_name>&time=<string:current_time>')
api.add_resource(IsTeacherHere, '/api/is_teacher_here/teacher=<string:teacher_name>')
api.add_resource(CreateDatabase, '/api/create_database')
api.add_resource(UpdateDatabase, '/api/update_database')
api.add_resource(GetTeachers, '/api/get_teachers')
api.add_resource(GetGroups, '/api/get_groups')
```

Рисунок 3.18 – модуль routes

Модуль lists_generator зтягує дані з Schedule API, виконує операцію парсингу і повертає асоціативний масив із списками даних (рис. 3.19).

```python
def get_generated_lists():

    request_groups = (grequests.get('http://api.pnu-bot.pp.ua/api/groups'),)
    groups = grequests.map(request_groups)[0].json()

    teachers_prev = []
    auds_prev = []
    subjects = []
    time_bounds_prev = []
    requests = (grequests.get(f'http://api.pnu-bot.pp.ua/api/schedule?group={group}&date_to=17.03.20') for group in groups)
    responses = grequests.map(requests)
    for response in responses:
        schedule = response.json() if response else {'schedule': []}
        if schedule['schedule']:
            group = schedule['group']
            items = schedule['schedule'][0]['items']
            for item in items:
                subject_num = int(item['number'])
                time_bound = item['time_bounds'].split(' - ')
                time_bounds_prev.append((time_bound[0], time_bound[1], subject_num))
                info = item['info'].split(' /  ')
                if info[0][0] != ' ':
                    aud = info[0]
                    n = 2 if '.' in info[1].split()[1] else 3
                    teacher = ' '.join(info[1].split()[:n])
                    name = info[2] if len(info) >= 4 else ' '.join(info[1].split()[n:])
                auds_prev.append(aud)
                teachers_prev.append(teacher)
                subjects.append((teacher, group, aud, subject_num, name))
    teachers_cur = list(dict.fromkeys(teachers_prev))
    auds_cur = [(a, i) for a in list(dict.fromkeys(auds_prev)) for i in range(1, 9)]
    time_bounds_cur = list(dict.fromkeys(time_bounds_prev))

    return {'teachers': teachers_cur,
            'groups': groups,
            'auds': auds_cur,
            'subjects': subjects,
            'time_bounds': time_bounds_cur
            }
```

Рисунок 3.19 – модуль lists_generator

Модуль subject_number_generator генерує номер навчальної пари з отриманого часу, використовує допоміжні функції get_time та get_time_bounds (рис. 3.20).

```python
def get_time(time_str):
    """
    :param time_str:        (string) time like `20:02`
    :return:                (object) converted string time to object time.struct_time
    """
    return datetime.strptime(time_str, '%H:%M')


def get_generated_subject_num(current_time):
    """
    Generates subject number by current time

    :param current_time:    (string) current time like `20:02`
    :return:                (int) generated subject number or None object
    """
    from project.models.functions import get_time_bounds
    time_bounds = get_time_bounds()
    if not time_bounds:
        return None
    subject_time = get_time(current_time)
    delta = timedelta(minutes=20)
    for time_bound in time_bounds:
        start_time = get_time(time_bound.start) - delta
        end_time = get_time(time_bound.end)
        if start_time <= subject_time < end_time:
            return time_bound.num
```

Рисунок 3.20 – модуль subject_number_generator

Модуль functions безпосередньо працює з об’єктами моделей власної бази даних для API (рис. 3.21).

```
def delete_all():
    db.session.query(Teachers).delete()
    db.session.query(Groups).delete()
    db.session.query(Auditoriums).delete()
    db.session.query(Subjects).delete()
    db.session.query(TimeBounds).delete()
    db.session.commit()


def get_subjects(current_time=None, group_name=None, teacher_name=None):
    """
    Generates `join` query of `subjects` and `auditoriums` tables

    :param current_time:        (string) string value of time in `%H:%M` format, like `20:20`
    :param group_name:          (string) group's name
    :param teacher_name:        (string) teacher's name
    :return:                    (list<Subjects, Auditoriums>) list of two filtered objects
    """
    query = db.session.query(Subjects, Auditoriums).join(Auditoriums, Subjects.aud_id == Auditoriums.id)
    result = query.filter((Subjects.group_id == get_group_id(group_name)) |
                          (Subjects.teacher_id == get_teacher_id(teacher_name)))
    if not current_time:
        return result
    subject_num = get_generated_subject_num(current_time)
    return result.filter(Auditoriums.subject_num == subject_num)


def get_free_auds(current_time):
    subject_num = get_generated_subject_num(current_time)
    query = Auditoriums.query.filter_by(subject_num=subject_num, status=False).all()
    return query


def create_db():
    db.create_all()


def update_db():
    # Dictionary of generated lists
    lists = get_generated_lists()

    delete_all()
    insert_teachers(lists['teachers'])
    insert_groups(lists['groups'])
    insert_auds(lists['auds'])
    insert_subjects(lists['subjects'])
    insert_time_bounds(lists['time_bounds'])
```

Рисунок 3.21 – Модуль functions

Модуль models ініціалізує моделі бази даних, використовує модуль SQLAlchemy (рис. 3.22).

```python
db = SQLAlchemy(app)


class Teachers(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    full_name = db.Column(db.String(40), nullable=False, unique=True)


class Groups(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    group_name = db.Column(db.String(20), nullable=False, unique=True)


class Auditoriums(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    aud = db.Column(db.String(30), nullable=False)
    subject_num = db.Column(db.Integer, nullable=False)
    status = db.Column(db.Boolean, nullable=False, default=False)


class Subjects(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    teacher_id = db.Column(db.Integer, db.ForeignKey('teachers.id'), nullable=False)
    group_id = db.Column(db.Integer, db.ForeignKey('groups.id'), nullable=False)
    aud_id = db.Column(db.Integer, db.ForeignKey('auditoriums.id'), nullable=False)
    name = db.Column(db.String(150), nullable=False)


class TimeBounds(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    start = db.Column(db.String(5), nullable=False)
    end = db.Column(db.String(5), nullable=False)
    num = db.Column(db.Integer, nullable=False)
```

Рисунок 3.22 – Модуль models

API даного проекту отримує дані про розклад від Schedule API, обробляє їх, та записує у власну базу даних. Після цього створює доступні URL адреси та обслуговує їх власними функціями, призначеними для зчитування даних з новоствореної бази.

4 ЕКОНОМІЧНА ЧАСТИНА ПРОЕКТУ

4.1 Аналіз ІТ-ринку України

Малий бізнес, сфера діяльності якого є інформаційні технології, доволі часто стикається із деякими трудностями.

Перша – полягає у витратах часу та ресурсів. Розробка програмного продукту потребує великих грошових вкладень і багато часу. Перед тим як передати готовий продукт споживачеві, він проходить кілька етапів: формування ідеї, розробку, тестування, презентацію продукту ринку. Таким чином, потрібно як мінімум кілька тижнів на запуск навіть нескладного проекту. Проте розраховувати на прибуток можна тільки після закінчення стадії розробки продукту.

Суть другої – робота з клієнтами. Кілька років тому клієнт не був готовий платити за послуги. Продавались лише програмні продукти, які обслуговувалися безкоштовно. Плюс цієї ситуації у тому, що можна було розробити програмний продукт і деякий час зосередити увагу на його продажі Мінус полягає в тому, що продажі не бували стабільними. Платіжна спосібність користувачів знижувалась за рахунок кілька факторів: неактуальність продукту, невідповідність бажанням чи потребам клієнта, необхідність у оновленні проекту та покупки нових версій, та багато інших.

Сьогодні ситуація змінилася. Велику кількість програмних продуктів можна отримати безкоштовно, завантаживши їх із сайтів виробників, при цьому обслуговування та налаштування програмного продукту під особисті потреби клієнта потребує коштів. Спосіб заробітку шляхом продажі послуг в ІТ сфері аналогічний до абонентської плати за послуги інтернет провайдера чи мобільного зв'язку. Замість покупки програмного продукту, актуальність якого буде на протязі 2-3 років, ціна якого наприклад 20 тис. грн., клієнт заплатить 2 тис грн. раз на рік і буде отримувати на постійній основі всі оновлення протягом проплаченого терміну, наприклад року. Така система вигідна клієнту

та компанії-розробнику, оскільки завжди легше піти на менші витрати, так як доволі рідкісним явищем є відмова від послуги, на відміну від відмови отримання цілого програмного продукту.

При погіршенні спроможності оплати, від послуги не відмовляються, а користуються рідше. Також нікому не спадає на думку проявляти незадоволення послугою і навіть вимагати повернення коштів. Крім того, легко планувати прибуток компанії завдяки таким факторам як: періодичність і систематичність оплати за послуги та збалансованість кількості споживачів так як одні користувачі перестають користуватись послугою, інші – починають, тобто з'являються нові клієнти.

Третя проблема полягає у підборі кадрів та тривалий період фінансової віддачі від IT-фахівців. В IT-сфері при підборі спеціаліста перед малим бізнесом стоять дві задачі: вірно визначити кваліфікацію спеціаліста та боротись із перекосами в очікуваннях співробітників стосовно зарплати.

Недостача розробників та присутність більшої частки зарубіжних компаній на українському IT-ринку (до 90% зарубіжних компаній), призвели до того, що зарплатні очікування співробітників, або кандидатів на різного роду посади в IT галузі, перекосились. Великі закордонні компанії доволі охоче інвестують у молодих, недосвідчених спеціалістів, які володіють навиками розробки програмних продуктів. Хоч ці інвестиції не завжди окуповуються, такий фахівець на випробувальному терміні може гарантовано отримати 1000-1500 доларів США заробітної плати. Зарубіжні компанії можуть собі таке дозволити оскільки для них українські спеціалісти є відносно дешевою, але водночас висококваліфікованою робочою силою. На Україну, як правило, припадають складні в архітектурному плані та плані розробки проекти але не надто великі. У свою чергу менш складні але об'ємні та рутинні проекти передаються азіатським відділенням, де порівняно з українськими спеціалістами ще нижче оцінюється робоча сила.

Таким чином, інженери, які працюють в галузі IT, порівняно з інженерами інших сфер діяльності, отримують відносно високу заробітну плату. Внаслідок цього зарплатні очікування зростають теж і фахівець, який прийшов на співбесіду та не може вирішити елементарні задачі для малого бізнесу, хоче отримати зарплату 500 доларів США як мінімум. Для виходу із цієї ситуації підприємці запрошують спеціалістів за рекомендацією, або приймають на роботу співробітників віком від 25-ти років і з досвідом роботи над комерційними проектами. Такі кандидати знають ціну своїх навиків та не завищують її в кілька разів. Від них ніхто не очікує надмірного росту кваліфікації. Фахівці це знають і розуміють. Крім цього, такі люди вже мають сім'ю або інші зобов'язання, вони в більшості зацікавлені стабільністю та надійністю, а не пошуком іншої роботи із зарплатою на декілька відсотків вище. У таких співробітників можна з більшою впевненістю інвестувати ніж у новачків.

У випадках, коли компанія зайнята розробкою програмного продукту для широкого кола користувачів, конкурентний тиск зазвичай високий, у випадках, коли займається розробкою під індивідуального клієнта – не відчувається. Доволі мала частка компаній займається розробкою індивідуальних замовлень, а такі продукти не можна повторити, так як вони завжди вимагають адаптацію під свої особисті потреби.

Згідно дослідження компанії N-IX IT-ринок України виріс у два рази та досяг більше 180 тисяч спеціалістів у даній сфері за останні чотири роки..

Одним з найбільш головних напрямків у сфері експорту послуг в Україні став IT-ринок оскільки дохід від нього сягає близько п’яти мільярдів доларів США на рік. Згідно звіту CEE станом на 2019 рік, напрям розробки програмних продуктів в Україні підвищився на 19 відсотків у 2018 році (рис. 4.1).

Рисунок 4.1 – Графік росту кількості IT працівників в залежності зі зростом IT ринку

Згідно даних організацій WorldBank та IT Ukraine, IT-ринок України виглядає таким чином:

- Україна має більше ніж півтори тисячі компаній, що займаються розробкою IT-послуг;
- Більше ніж 185 000 спеціалістів в даній сфері;
- Більше ста компаній в списку Fortune, який представляє собою рейтинг топових компаній світу, є клієнтами IT-фірм України;
- ІКТ (Інформаційно-комунікаційні технології) є третьою за величиною галуззю у сфері експорту послуг України, частка якої сягає п’яту частину всієї сфери.

Згідно з опитування українських IT-спеціалістів виявленоо, що біля 67% з них проживають в Львові, Києві та Харкові.

На інтернет-порталі “Stack Overflow” вказано що, в місті Києві знаходиться близько 40% IT-фахівців (рис. 4.2).

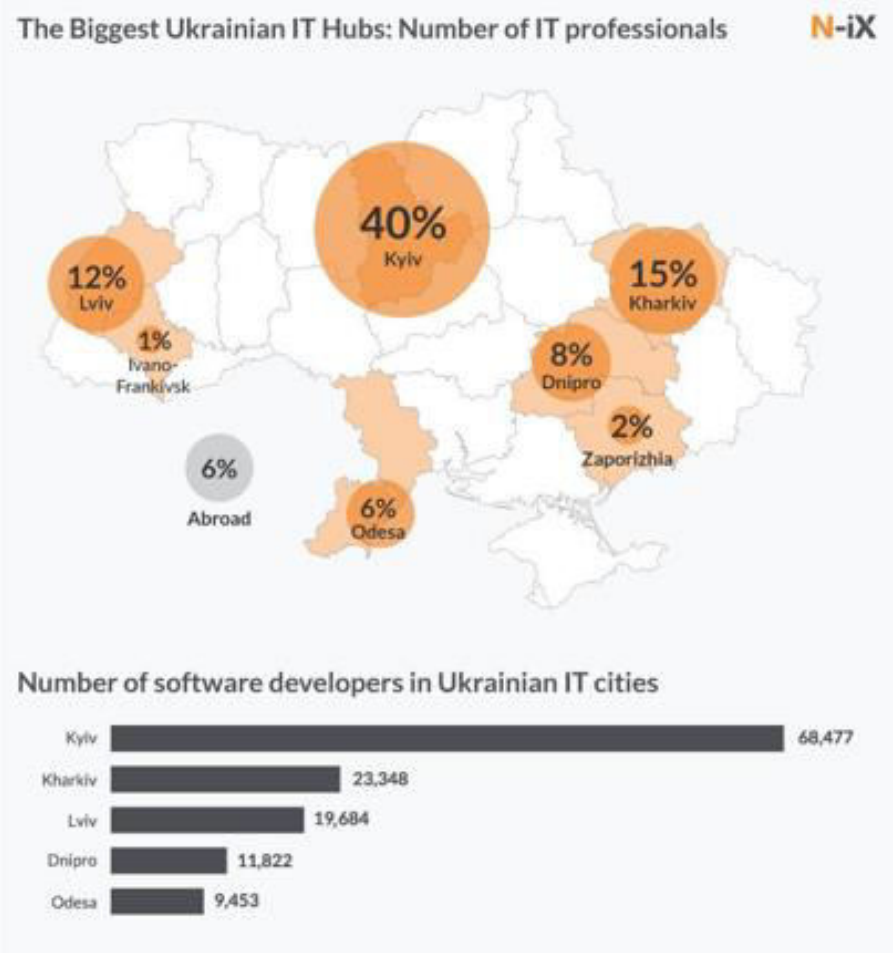

Рисунок 4.2 – Діаграма кількість IT-фахівців в Україні

Другим IT-центром України за величиною є місто Львів. Там працюють більше 20 тисяч IT-спеціалістів і біля 317 IT-компаній, більшу долю яких складають компанії, що займаються розробкою програмних продуктів для клієнтів у Західній Європі та Сполучених Штатах Америки.

4.2 Аналіз економічної частини проекту

Розрахуємо середню годинну оплату програміста. Для цього необхідно спочатку визначити його річний фонд грошового забезпечення.Це можна зробити, знаючи місячне грошове забезпечення програміста. Воно складає приблизно 24000,00 гривень (В роботі, взяте середнє значення грошового забезпечення програміста за 2020 рік). Таким чином, річний фонд грошового забезпечення 288000 гривень. Визначимо число робочих годин у році. Для цього від кількості днів у році потрібно відняти загальну кількість вихідних та святкових днів і отриманий результат помножити на 8 (на кількість робочих годин). У році 365 днів, з яких 10 – святкові, 104 – вихідні. Отже, кількість робочих годин у році дорівнює добутку 341 на 8, що в результаті дорівнює 2008 години.

Середня годинна оплата програміста визначається співвідношенням річного фонду грошового забезпечення до річної кількості робочих годин. Результатом буде частка 288000 грн від 2008-ми годин, тобто 143,43 гривень.

На розробку даного проекту було затрачено близько 28 робочих днів, тобто 224 робочі години. Оплата за проект загалом виходить як добуток 143,43 грн/год та 224 год, тобто 32 128,32 грн.

ВИСНОВКИ

Близько 30-ти днів було потрачено на розробку даного проекту. Проект розбитий на Front-end та Back-end частини.

Front-end частиною проекту є голосовий асистент у вигляді Android-додатку. Додаток дає користувачу можливість авторизуватись, щоб не вказувати при кожному використанні додатку свої дані, такі як прізвище та ініціали якщо являється викладачем або назву групи якщо являється студентом. Крім цього додаток дозволяє відправити запит на back-end проекту з допомогою голосових команд.

Back-end частина проекту – API, назване DekanatAPI, розроблене на мові програмування Python 3, фреймворку Flask та його модулі FlaskRESTful.

API проекту надає можливість проводити складні запити, типу як дізнатись список вільних аудиторій у даний проміжок часу, чи навіть забронювати вільну аудиторію, чого не можуть дати аналоги проекту.

Значущість проекту доволі не мала для університету, а точніше для студентів та викладачів, так як дозволяє отримати складну інформацію вже і зараз. Цей проект можна використовувати і за межами університету, але однозначно оприділити те що користувачі будуть використовувати проект не тільки в межах університету – неможливо.

Можливість застосування проекту доволі велика, так як не вимагає багато затрат ресурсів як фізичних так і фінансових. Проте існує імовірність що від додатку університет відмовиться, а API буде використовувати та розвивати, так як воно більш універсальне і може працювати на будь-яких платформах. Вся суть проекту полягає якраз у API, основна логіка проекту знаходиться тут, а Android-додаток – це один із способів подачі інформації яку відображає API.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Що таке віртуальний асистент. URL: http://ipkey.com.ua/uk/faq/936-virtual-assistant.html (дата звернення: 10.04.2020)
2. Яким буде 2019 рік для голосових асистентів. URL: https://news.samsung.com/ua/2019-for-voice-assistants?cid=in_paid_display_affiliate_cake_cefest_eshop_hhp_20200101_none (дата звернення: 10.04.2020)
3. Голосові асистенти: що вони вміють і чим відрізняються один від одного.URL: https://aiconference.com.ua/uk/news/golosovie-assistenti-chto-oni-umeyut-i-chem-otlichayutsya-drug-ot-druga-97306 (дата звернення: 10.04.2020)
4. Alexa, Duplex, Siri та інші: куди рухаються технології голосових помічників? URL: https://www.imena.ua/blog/alexa-duplex-siri-development/ (дата звернення: 10.04.2020)
5. Корпоративний блог компанії “Яндекс”. 10.10.2017. URL: https://yandex.ru/blog/company/alisa (дата звернення: 10.04.2020)
6. Microsoft'svirtualassistant, Cortana — TheNewYorkTimes. URL: https://www.nytimes.com/2015/07/30/technology/personaltech/windows10-review.html (дата звернення: 10.04.2020)
7. CortanaAnalyticsSuiteOverview. URL: https://www.microsoft.com/en-us/server-cloud/cortana-analytics-suite/overview.aspx

(дата звернення: 10.04.2020)
8. Український IT-ринок. URL:
http://yasno-group.com/ua/проекти/думка_експерту/український_it-ринок/
(дата звернення: 21.05.2020)
9. Який наразі в Україні IT-ринок: аутсорс, кількість працівників та компаній. URL:
https://nachasi.com/2019/11/07/it-ukraine/
(дата звернення: 21.05.2020)
10. Інформаційний вузол спільної Робочої групи з питань реорганізації системи управління домену. URL:
http://ua-nic.net/
(дата звернення: 21.05.2020)
11. Правила реєстрації доменних імен в домені ORG.UA. URL:
http://org.ua/
(дата звернення: 21.05.2020)
12. Безкоштовні домени — опис процедури безкоштовної реєстрації деяких українських доменів. URL:
http://freedom.org.ru/ukr.php
(дата звернення: 21.05.2020)
13. Рейтинг українських реєстраторів доменів. URL:
http://tops.org.ua/uk/reytynh-ukrayinskykh-reyestratoriv-domeniv.html
(дата звернення: 21.05.2020)
14. Intents and Intent Filters. Android Developers. URL:
https://developer.android.com/guide/components/intents-filters?hl=ua
(дата звернення: 21.05.2020)
15. Toasts overview. Android Developers. URL:
https://developer.android.com/guide/topics/ui/notifiers/toasts
(дата звернення: 21.05.2020)
16. Офіційний сайтSQLite. URL:

http://www.sqlite.org/
(дата звернення: 21.05.2020)
17. SQLiteDatabaseBrowser — візуальний засіб адміністрування SQLite. URL:
http://sqlitebrowser.sourceforge.net/
(дата звернення: 21.05.2020)
18. SQLiteAdministrator — засіб адміністрування і редактор SQL-запитів в SQLite. URL:
http://sqliteadmin.orbmu2k.de/
(дата звернення: 21.05.2020)
19. RFC 2616. Стандарт HTTP/1.1. URL:
https://tools.ietf.org/html/rfc2616
(дата звернення: 21.05.2020)
20. RFC 2818. HTTP Over TLS. URL:
https://tools.ietf.org/html/rfc2818
(дата звернення: 21.05.2020)
21. Мікро-фреймворк Flask. URL:
https://pypi.python.org/pypi/Flask
(дата звернення: 21.05.2020)
22. Release 1.1.2. URL:
https://github.com/pallets/flask/releases/tag/1.1.2
(дата звернення: 21.05.2020)
23. Volley overview. Android Developers. URL:
https://developer.android.com/training/volley
(дата звернення: 21.05.2020)
24. Intent. Android Developers. URL:
https://developer.android.com/reference/android/content/Intent?hl=ua
(дата звернення: 21.05.2020)