ЗМІСТ ОТ ТУТ

ВСТУП

Залізничний транспорт є однією з найбільш важливих складових транспортно-дорожнього комплексу. Він відіграє значну роль у забезпеченні життєдіяльності багатогалузевої економіки України; на його долю припадає 82% вантажоперевезень та майже 40% пасажирських перевезень, що здійснюються усіма видами транспорту.

Пасажири обирають залізничний транспорт здебільшого через його надійність, високу швидкість та відносно низьку вартість проїзду (порівняно, наприклад, із автомобільним транспортом). Але дуже важливим критерієм також є зручність користування.

У мережі Інтернет уже давно можна знайти розклад руху потягів, дізнатися про вартість квитка та навіть замовити квиток (наприклад, за допомогою сайту Укрзалізниці, сервісу «Приват 24» та ін.). Для подібних цілей існують також мобільні додатки (зокрема, той самий «Приват 24» для Android). Однак і на сайтах, і в додатках для мобільних пристроїв використовуються стандартні на даний момент засоби введення інформації – клавіатура, маніпулятор «миша», сенсорний екран.

Розроблювана система розпізнання голосових запитів для пошуку залізничних квитків та супутньої інформації має схожий функціонал та сферу застосування (придбання квитків), однак надає змогу задавати критерії пошуку та вводити основні команди керування програмою за допомогою голосу. Такий спосіб введення інформації є більш природним для людини, а отже – більш зручним. Наразі аналогів (програм з аналогічними функціями та способом введення даних) розроблюваної системи не існує.

Впровадження даного програмного засобу дасть змогу залучити до сервісів Укрзалізниці більше клієнтів (здебільшого за рахунок користувачів мобільних платформ), а також – у перспективі – отримувати прибуток завдяки монетизації додатку.

1 ЗБІР ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1.1 Аналіз літературних джерел

При проектуванні та розробці програмного засобу використовувались перераховані нижче літературні джерела.

1.1.1 Радзишевский А. Ю. «Основы аналогового и цифрового звука»

У книзі розглядається широкий спектр питань, пов’язаних зі звуком: від основ теорії звуку до детального опису існуючих способів цифрового представлення звуку і базисних методів його обробки і стиснення.

Із даної книги було взято інформацію про структуру основних звукових форматів, а також алгоритм розрахунку амплітуди звуку для визначення рівня гучності записуваного звуку (за допомогою цього можна розпізнати, коли користувач закінчив говорити).

1.1.2 Крейг Уоллс «Spring в действии» (Craig Walls «Spring in action»)

Книга містить опис третьої версії фреймворку Spring, зокрема інформацію про мову виразів SpEL, нові анотації для робот из контейнером IoC і підтримку архітектури REST.

Книга містить значну кількість прикладів коду, які сприяли розумінню принципів роботи Spring MVC та шаблону проектування Model-View-Controller в цілому.

1.1.3 Дэвид Флэнаган «JavaScript. Подробное руководство» (David Flanagan «JavaScript. The Definitive Guide»)

У книзі, окрім основ мови, розглядається взаємодія із протоколом HTTP, застосування технології AJAX, обробка документів XML та JSON, середовище розробки сценаріїв у браузерах та ін. Книга містить численні приклади коду та обширний довідник із базового та клієнтського JavaScript (класи, методи, властивості, константи і т.д.)

Книга використовувалася здебільшого саме як довідник при розробці клієнтської частини додатку.

1.1.4 Б. Фрайн «HTML5 и CSS3. Разработка сайтов для любых браузеров и устройств» (B. Frain «Responsive Web Design with HTML5 and CSS3»)

Книга описує методологію адаптивного веб-дизайну (тобто такого, що враховує можливості відображення даних та взаємодії із користувачем на різних платформах – як уже існуючих, так і тих, що з’являться найближчим часом).

Із даної книги, окрім деяких базових знань про HTML5 та CSS3, було взято методи відображення сторінок залежно від розміру екрана, оскільки передбачається, що розроблюваний додаток функціонуватиме не лише на персональних комп’ютерах, а і на мобільних пристроях.

1.1.5 Амит Бакор «Apache Tomcat для профессионалов» (Amit Bakore «Professional Apache Tomcat»)

Дана книга описує процес встановлення та налаштування Tomcat, його інтегрування з Apache, налаштування та підключення до бази даних за допомогою драйвера JDBC, захист програм. Також обговорюються деякі додаткові інструменти, зокрема Ant (для автоматичного компонування веб-додатків).

Із цієї книги взято відомості про методи та програмні засоби, необхідні для розгортання розроблюваного додатку на сервері.

1.2 Аналіз програмних аналогів

Оскільки програм, повністю ідентичних за функціоналом до розроблюваної системи, немає, натомість були окремо проаналізовані голосові помічники та засоби замовлення квитків. Розглянуті програми описані нижче.

1.2.1 Google Now

Google Now – персоналізований сервіс пошуку від Google Inc, що використовує обробку природної мови для відповіді на запитання, створення рекомендацій і виконання різноманітних дій. Додаток видає інформацію з урахуванням поточного місцезнаходження користувача, його особистої інформації з календаря, історії пошукових запитів, історії переміщень, історії відвіданих сторінок та ін. Результати оформлені у вигляді так званих «інформаційних карток», кожна з яких відповідає за свій тип інформації (погода, трафік, зустрічі, авіарейси, спорт, місця, переклад, валюта та ін.). Запущений 9 липня 2012 року. Зовнішній вигляд Google Now відображено на рис. 1.1.



Рисунок 1.1 – Зовнішній вигляд Google Now

Переваги:

– інтерфейс, зручний для постійного оновлення інформації;

– підтримка значної кількості мов, у тому числі російської;

– широкий спектр функцій: управління функціями телефону (набір номерів, відправлення повідомлень, створення нагадувань і т. ін.), функції пошуку (прогноз погоди, встановлення маршрутів), голосове введення тексту;

– інтеграція з іншими сервісами Google (Gmail, Maps, Translate та ін.);

– сумісність із Google Glass.

Недоліки:

– доступний лише у вигляді додатка для пристроїв на Android та iOS;

– не реагує на нестандартні запити.

1.2.2 Siri

Siri – персональний помічник і питально-відповідна система, адаптована для iOS. Цей додаток використовує обробку природної мови, щоб відповідати на питання і давати рекомендації. Siri пристосовується до кожного користувача індивідуально, вивчаючи його переваги протягом довгого часу. Є розробкою Міжнародного Центру Штучного Інтелекту SRI. Запущена 9 серпня 2011 року. Зовнішній вигляд Siri відображено на рис. 1.2.

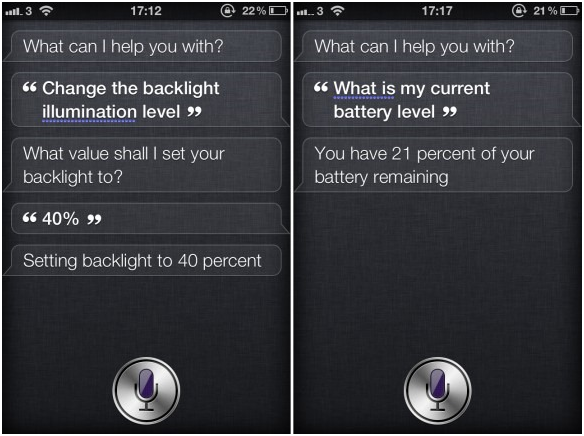


Рисунок 1.2 – Зовнішній вигляд Siri

Переваги:

– розпізнання змісту нетипових запитів, включаючи залежні від контексту;

– підтримка значної кількості мов, у тому числі російської;

– широкий спектр функцій: управління функціями телефону (здійснення дзвінків, зміна налаштувань тощо), взаємодія з іншими додатками (музика, годинник, браузер, карти, пошта та ін.), здійснення операцій в Apple Store, голосове введення тексту;

– якісний голос (у двох варіантах);

– взаємодія із сервісами OpenTable, ReverseTravel, Yahoo, Bing, StubHub, Rotten Tomatoes, The New York Times, Wolfram Alpha та ін.

Недоліки:

– доступна виключно як інтегрована частина iOS;

– підтримка російської мови на даний момент реалізована недостатньо якісно.

1.2.3 Cortana

Cortana – віртуальний голосовий помічник зі штучним інтелектом для пристроїв на Windows Phone. Призначений передбачати потреби користувача. Має вікове обмеження – доступний лише користувачам віком не менше 13 років. Запущений 2 квітня 2014 року. Буде інтегрований у Windows 10. Зовнішній вигляд Cortana відображено на рис. 1.3.

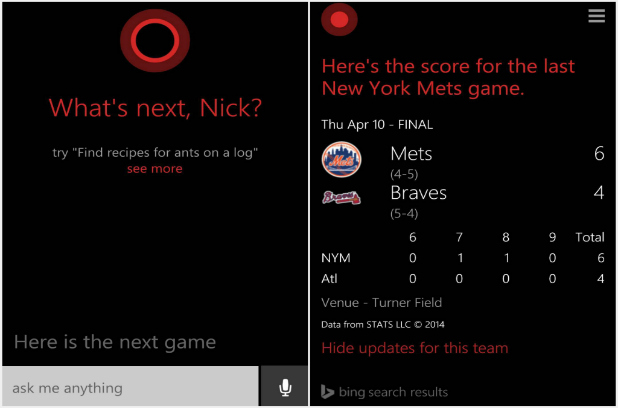


Рисунок 1.3 – Зовнішній вигляд Cortana

Переваги:

– можливість спілкування на вільні теми;

– відстеження інтересів користувача;

– гнучкі налаштування конфіденційності (можливість указати, яку саме інформацію помічникові дозволено збирати);

– широкий спектр функцій: управління функціями телефону (дзвінки, повідомлення, нагадування та ін.), функції пошуку, голосове введення тексту.

Недоліки:

– доступний лише для пристроїв на Windows Phone;

– немає підтримки російської мови.

1.2.4 Laitis

Laitis – програма для набору тексту та управління комп’ютером за допомогою голосу. Зовнішній вигляд Laitis відображено на рис. 1.4.

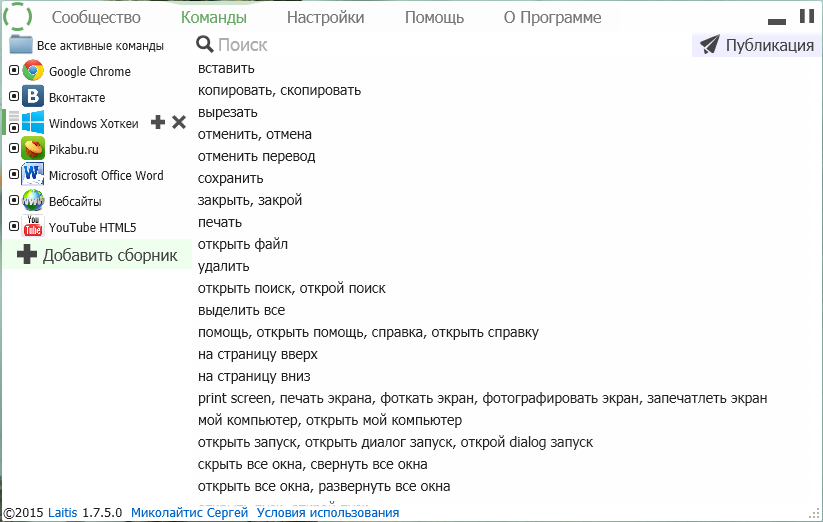


Рисунок 1.4 – Зовнішній вигляд Laitis

Переваги:

– можливість набору тексту голосом зі збереженням пунктуації, коректним відображенням власних назв та іншомовних слів;

– підтримка російської мови;

– можливість запуску програм голосом;

– можливість створення власних шаблонів (можна призначити голосову команду для запуску програми, натиснення комбінації клавіш тощо).

Недоліки:

– програма працює лише на системах, де встановлений .NET Framework.

1.2.5 RuVC

RuVC (Russian Voice Control) – програма для керування комп’ютером (запуску програм, швидкого пошуку, набору тексту, пошуку в мережі) за допомогою голосу. Зовнішній вигляд RuVC відображено на рис. 1.5.

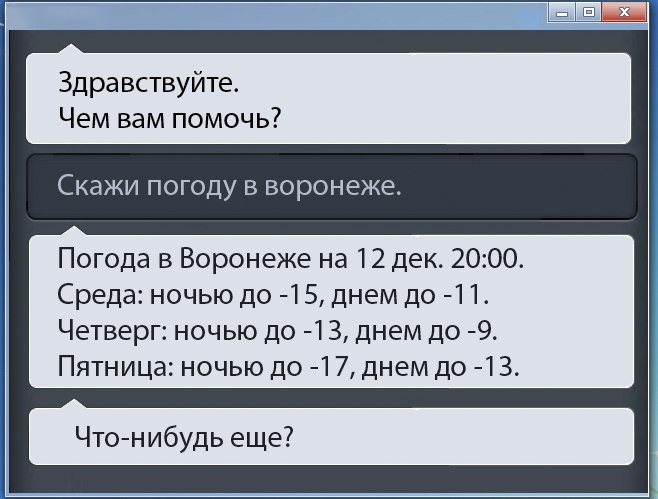


Рисунок 1.5 – Зовнішній вигляд RuVC

Переваги:

– підтримка російської мови;

– можливість запуску програм голосом;

– можливість створення власних шаблонів;

– програма супроводжує виконання дій голосовими коментарями.

Недоліки:

– програма працює лише на системах, де встановлений .NET Framework;

– супровід програми тимчасово призупинений.

1.2.6 Приват 24

Приват 24 ¬ система інтернет-банкінгу, яка, серед інших функцій, надає можливість замовляти залізничні квитки онлайн. Зовнішній вигляд системи Приват 24 відображено на рис. 1.6.

Переваги:

– значна кількість налаштувань та їх наглядне відображення (можна обрати пункти відправлення та прибуття, дату відправлення, потяг, вагон і місце, додатково можна дізнатися приблизний час у дорозі, тип вагонів, кількість вільних місць та вартість квитка);

– можливість замовити до чотирьох квитків одразу, а також замовити квиток на потяг у зворотній бік;

– надійність (використовується двохетапна аутентифікація, сучасні протоколи шифрування та перевірений сертифікат безпеки);

– зручність оплати замовлених квитків;

– наявність додатку для мобільних пристроїв.

Недоліки:

– для користування послугами системи необхідна реєстрація та авторизація;

– система доступна тільки для клієнтів Приватбанку.

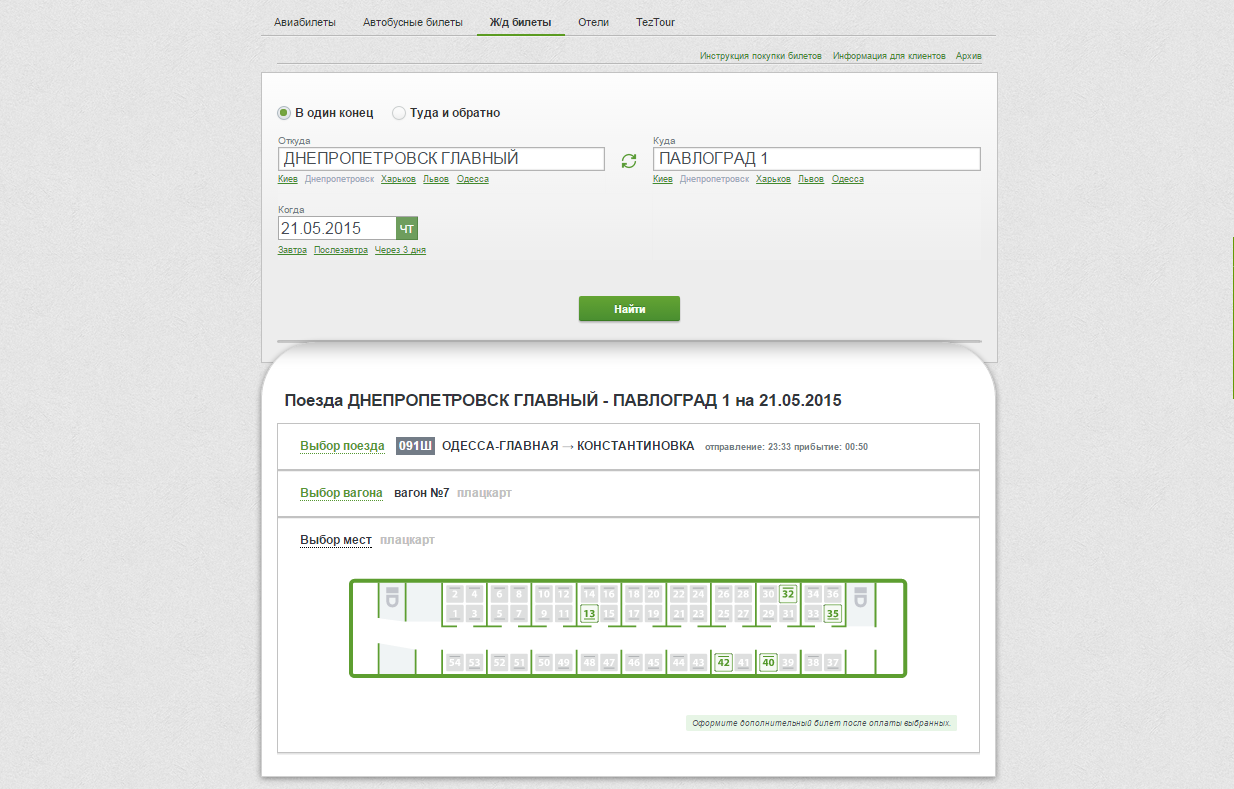


Рисунок 1.6 – Зовнішній вигляд системи Приват 24

1.2.7 Замовлення квитків на сайті Укрзалізниці

На офіційному сайті Укрзалізниці можна замовити квитки онлайн. Зовнішній вигляд сайту відображено на рис. 1.7.

Переваги:

– значна кількість налаштувань та їх наглядне відображення (можна обрати пункти відправлення та прибуття, дату відправлення, потяг, вагон і місце, додатково можна дізнатися приблизний час у дорозі, тип вагонів, кількість вільних місць та вартість квитка);

– можливість замовити квиток на потяг у зворотній бік;

– можливість замовлення квитків без реєстрації.

Недоліки:

– система довгий час знаходиться на етапі публічного тестування та не оновлюється.

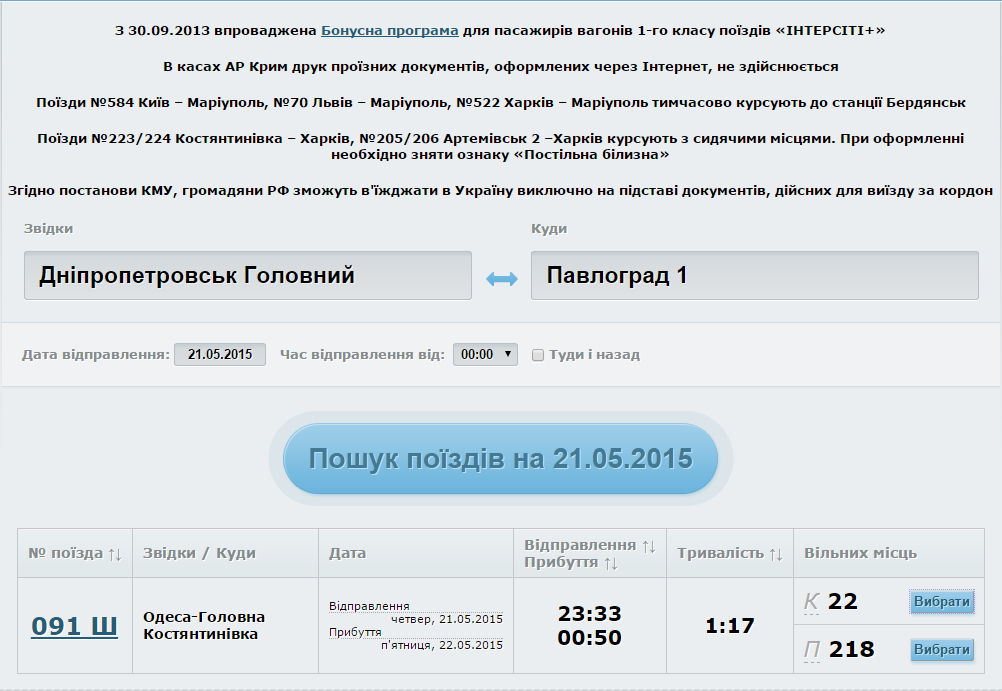


Рисунок 1.7 – Зовнішній вигляд сайту Укрзалізниці

1.2.8 Аналіз аналогів та висновки

На сучасному ринку відсутні програмні засоби із функціоналом, аналогічним до того, який матиме система розпізнання голосових запитів для пошуку залізничних квитків. Тому в цьому розділі був розглянутий більш загальний випадок – додатки, які виконують функції голосових помічників. Перші три з них (Google Now, Siri та Cortana) розроблені для використання на мобільних пристроях, мають широкий спектр можливостей для застосування і достатньо широко розповсюджені.

Наступні два (Laitis та RuVC), розроблені для використання на персональних комп’ютерах, мають досить обмежений функціонал (оскільки створенням кожної з цих програм займалась одна людина) та не користуються значною популярністю.

Також були розглянуті системи, що дозволяють знайти та замовити залізничний квиток. Сайти Укрзалізниці та Приват 24 виконують ті самі функції, що і розроблюваний програмний засіб, однак вони використовують стандартні методи введення (клавіатура та маніпулятор «миша»).

Можна зробити висновок, що розробка системи розпізнання голосових запитів для пошуку залізничних квитків є доцільною, оскільки:

– наразі не існує її повних аналогів;

– формат розробки (веб-додаток) сприятиме набуттю популярності системи, оскільки вона працюватиме на всіх платформах;

– метод голосового введення даних буде більш зручним для користувачів мобільних пристроїв;

– програма розпізнаватиме запити, зроблені російською мовою.

1.3 Опитування зацікавлених сторін

Опитування можна розглядати як один з найпоширеніших методів отримання інформації про суб'єктів – респондентів опитування. Опитування полягає в ставленні людям спеціальних питань, відповіді на які дозволяють досліднику отримати необхідні відомості залежно від завдань дослідження.

Для опитування було обрано метод анкетування.

Анкетування – метод опитування за допомогою анкет (структурно-організованих запитань, кожне з яких пов’язане з метою та завданнями дослідження). Анкета – впорядкований за змістом і формою набір запитань, що має вигляд опитувального листа.

Зацікавлені сторони – це особи чи організації, які мають дійсний інтерес до системи. Вона може впливати на них прямо чи опосередковано. До зацікавлених сторін можна віднести потенційних користувачів системи та адміністраторів. Користувачами системи можуть бути люди, які користуються послугами Укрзалізниці, володіють російською мовою та навичками користування комп’ютером або мобільним пристроєм. Адміністратори – люди, котрі мають спеціальні знання на займатимуться встановленням, налаштуванням і підтримкою серверної частини програмного засобу.

Дані, отримані у процесі опитування, наведені нижче.

1.3.1 Опитування користувачів

Метод опитування користувачів – анкетування.

Анкети містили питання стосовно побажань щодо можливостей, деталей роботи та інтерфейсу програми.

У результаті проведення опитування були виявлені наступні вимоги:

– простий мінімалістичний, але інформативний інтерфейс;

– оперативність роботи (хороша оптимізація на різних платформах);

– адекватні системні вимоги;

– можливість редагувати введені дані, скасовувати останні дії та керувати програмою за допомогою голосу;

– дублювання вихідної інформації (у вигляді тексту на екрані та усних фраз).

1.3.2 Опитування адміністраторів

Метод опитування адміністраторів – анкетування.

Анкети містили питання стосовно побажань щодо технічних характеристик серверної частини програми.

У результаті проведення опитування були виявлені наступні вимоги:

– встановлення на веб-сервер зі статичною IP-адресою (бажано із зареєстрованим доменним іменем);

– встановлення на веб-сервері сертифікату SSL для забезпечення обміну інформацією з користувачами через захищений протокол HTTPS;

– захищеність від SQL-ін’єкцій.

1.4 Специфікація вимог

У ході аналізу літературних джерел, аналогів та опитування зацікавлених осіб були виявлені наступні функціональні вимоги:

– дозволяти голосом обрати критерії для пошуку квитка (такі як місце відправлення та прибуття, дату відправлення та ін.);

– надання вибору конкретного квитка зі списку тих, що задовольняють указаним критеріям (якщо таких квитків декілька);

– виконання переадресації на сторінку придбання знайденого квитка;

– надання можливості у подальшому підтвердити факт придбання (шляхом роздрукування квитка або збереження коду транзакції);

– функціонування на мобільних платформах (також адаптація зовнішнього вигляду для коректного відображення на різних пристроях);

– можливість редагувати введені дані, скасовувати останні дії та керувати програмою за допомогою голосу;

– дублювання вихідної інформації (у вигляді тексту на екрані та усних фраз).

У ході аналізу літературних джерел, аналогів та опитування зацікавлених осіб були виявлені наступні нефункціональні вимоги:

– простий та зручний інтерфейс;

– швидкий відклик на дії користувача;

– захищеність системи.

2 ПРОЕКТУВАННЯ ПРОГРАМНОГО КОМПЛЕКСУ

2.1 Засоби проектування

Для проектування системи була обрана мова UML.

UML (англ. Unified Modeling Language) — уніфікована мова моделювання, використовується у парадигмі об'єктно-орієнтованого програмування. Є невід'ємною частиною уніфікованого процесу розробки програмного забезпечення. UML є мовою широкого профілю, це відкритий стандарт, що використовує графічні позначення для створення абстрактної моделі системи, названої UML-моделлю. UML був створений для визначення, візуалізації, проектування й документування в основному програмних систем. UML не є мовою програмування, але в засобах виконання UML-моделей як інтерпретованого коду можлива кодогенерація.

Діаграми UML дають можливість представити систему (як ділову, так і програмну) у такому вигляді, щоб її можна було легко перевести в програмний код. UML спеціально створювалася для оптимізації процесу розробки програмних систем, що дозволяє збільшити ефективність їх реалізації у кілька разів і помітно поліпшити якість кінцевого продукту. Діаграми підвищують супроводжуваність проекту і полегшують розробку документації.

В UML використовується 13 видів діаграм. Серед них:

– структурні (класів, компонент, композитної/складеної структури, кооперації, розгортання, об’єктів, пакетів);

– діаграми поведінки (діяльності, скінченних автоматів (станів), прецедентів);

– діаграми взаємодії (кооперації/комунікації, огляду взаємодії, послідовності, синхронізації).

У якості програмного засобу для створення діаграм UML було обрано пакет Rational Rose.

2.2 Діаграма прецедентів

Діаграма прецедентів (англ. Use Case Diagram, діаграма варіантів використання) відображає відносини між акторами та прецедентами і є складовою частиною моделі прецедентів, що дає змогу описати систему на концептуальному рівні.

Актори – ролі користувача (у широкому значенні: людина, зовнішня сутність, клас, інша система), який взаємодіє із іншими сутностями.

Прецеденти – можливості модельованої системи (частини її функціональності), завдяки яким користувач може отримати конкретний, вимірний і потрібний йому результат. Прецедент відповідає окремому сервісу системи, визначає один із варіантів її використання та описує типовий спосіб взаємодії користувача із системою.

Проектування ПЗ доцільно розпочинати саме зі створення діаграми прецедентів, оскільки вона описує функціональність і поведінку, які мають бути визначені найперше.

Для розроблюваної системи розпізнання голосових запитів для пошуку залізничних квитків та супутньої інформації було створено діаграму прецедентів, відображену на рис. 2.1.

Передбачається, що із системою взаємодіятиме тільки один актор – користувач. Він матиме можливість обрати пункт відправлення, пункт прибуття, дату відправлення. Після цього користувач зможе вибрати один зі знайдених за введеними параметрами варіантів потягів та перейти безпосередньо до придбання квитка.

Варіанти використання мають виконуватись по черзі (не можна, наприклад, ввести пункт прибуття, якщо не визначено пункт призначення). Однак, якщо користувача не задовольняють результати пошуку (або з інших причин), можна почати введення даних спочатку.

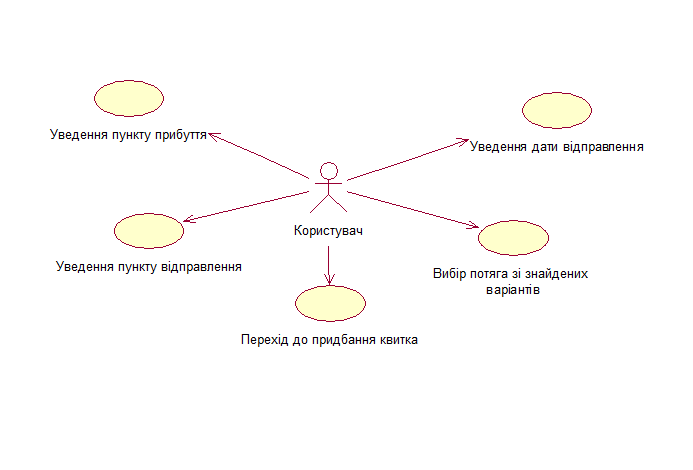


Рисунок 2.1 – Діаграма прецедентів

2.3 Проектування класів за допомогою CRC-карток

CRC-картка (англ. Class-responsibility-collaboration card; укр. картка «Клас-відповідальність-кооперація») – метод, призначений для проектування об’єктно-орієнтованого програмного забезпечення. CRC-картки використовуються у тих випадках, коли спочатку у процесі проектування ПЗ визначаються класи і способи їх взаємодії.

Зміст CRC-карток:

* назва класу;
* обов’язки класу;
* назви інших класів, з якими пов'язаний даний клас.

Використання CRC-карток при проектуванні ПЗ дає змогу мінімізувати складність дизайну, акцентувати увагу на сутностях (абстрагуватися від деталей), не призначати класу надто багато обов’язків.

У табл. 2.1 – 2.4 наведено розроблені CRC-картки для основних класів.

Таблиця 2.1 – CRC-картка для класу Entity

|  |  |
| --- | --- |
| Entity | |
| Обов’язки | Кооперація |
| Клас описує залізничний рейс | Використовується классом Server |

Таблиця 2.2 – CRC-картка для класу Server

|  |  |
| --- | --- |
| Server | |
| Обов’язки | Кооперація |
| Клас слугує для обробки запитів зі сторони клієнта та роботи із базою даних | Використовує клас Entity, взаємодіє із класом Client |

Таблиця 2.3 – CRC-картка для класу Client

|  |  |
| --- | --- |
| Client | |
| Обов’язки | Кооперація |
| Клас відповідає за взаємодію із сервером та відображення результатів роботи на web-сторінці | Взаємодіє із класом Server |

Таблиця 2.4 – CRC-картка для класу Recognizer

|  |  |
| --- | --- |
| Client | |
| Обов’язки | Кооперація |
| Клас займається розпізнанням мовлення та перетворенням голосових запитів на текст | Взаємодіє із класом Client |

2.4 Діаграма класів

Діаграма класів (англ. Static Structure Diagram) – діаграма, що демонструє класи системи, їх атрибути, методи і взаємозв’язки між ними.

Важливою складовою діаграми класів є взаємозв’язки між класами. Взаємозв’язок – це особливий тип логічних відношень між сутностями. В UML представлені наступні види відношень:

– асоціація (об’єкти одного класу (сутності) пов’язані з об’єктами іншої сутності);

– агрегація (об’єкт містить у собі екземпляри інших класів);

– композиція (більш суворий варіант агрегації; є жорстка залежність часу існування екземплярів класу-контейнера та об’єктів, що містяться в них);

– узагальнення, або наслідування (один клас (підтип) є окремою формою іншого (надтип), який називається узагальненням першого);

– реалізація (один елемент реалізує поведінку, задану іншим);

– залежність (зміни у специфікації одного класу ведуть до змін у специфікації іншого).

Для розроблюваної системи розпізнання голосових запитів для пошуку залізничних квитків та супутньої інформації було створено діаграму класів, відображену на рис. 2.2.

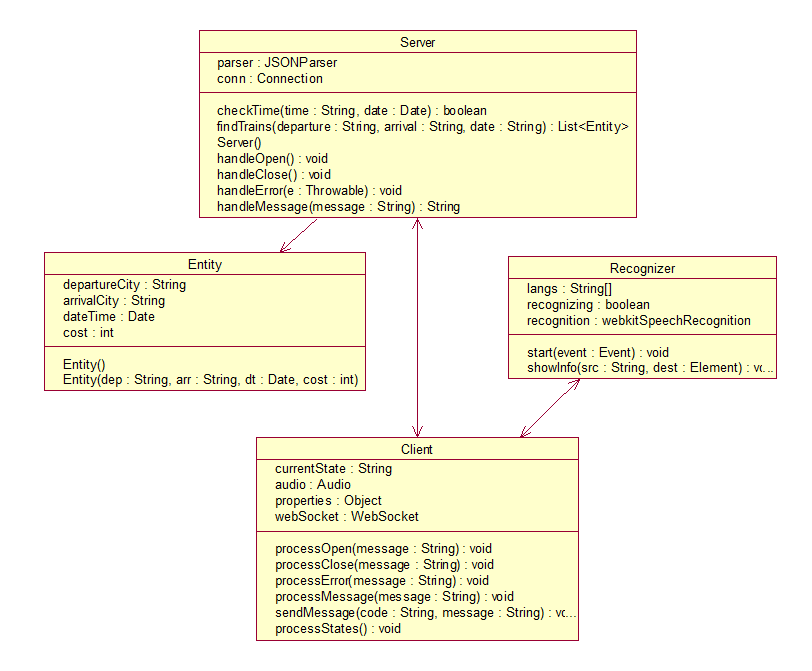


Рисунок 2.2 – Діаграма класів

1. Клас Entity

Клас описує залізничний рейс.

Поля:

– departureCity – пункт відправлення (у вигляді рядка);

– arrivalCity – пункт призначення (у вигляді рядка);

– dateTime – дата і час відправлення (у вигляді java.util.Date);

– cost – вартість квитка (у вигляді цілого числа).

Методи:

– Entity () – конструктор за замовчуванням;

– Entity (String dep, String arr, Date dt, int cost) –конструктор із параметрами;

– також клас містить методи отримання та встановлення значення для кожного із полів (так звані get- і set-методи).

1. Клас Server

Клас слугує для обробки запитів зі сторони клієнта та роботи із базою даних.

Поля:

– parser – об’єкт org.json.simple.parser.JSONParser, слугує для роботи з даними у форматі JSON;

– conn – об’єкт java.sql.Connection, необхідний для роботи з базою даних MySQL за допомогою драйвера JDBC.

Методи:

– checkTime (String time, Date date) – перевіряє коректність уведеної дати відправлення (якщо параметр time містить місяць і число, повертає true і змінює поточний стан параметра date, інакше повертає false);

– findTrains (String departure, String arrival, String date) – здійснює пошук рейсів із пункту відправлення departure до пункту призначення з датою відправлення date (повертає знайдені рейси у вигляді списку об’єктів Entity);

– Server () – конструктор за замовчуванням;

– handleOpen () – обробляє відкриття з’єднання із клієнтом;

– handleClose () – обробляє закриття з’єднання із клієнтом;

– handleError (Throwable e) – обробляє помилки при встановленні з’єднання (параметр е типу java.lang.Throwable містить інформацію про подію, що викликала помилку);

– handleMessage (String message) – обробляє повідомлення, що надходять від клієнта (містяться у параметрі message) та повертає власні повідомлення.

1. Клас Client

Клас відповідає за взаємодію із сервером та відображення результатів роботи на web-сторінці.

Поля:

– currentState – поточний стан програми (етап, на якому вона знаходиться: підтвердження пункту відправлення, уведення дати і т.п.);

– audio – звуковий запис голосової відповіді на запит;

– properties – об’єкт, що описує рейс;

– webSocket – об’єкт, що відповідає за встановлення з’єднання із сервером.

Методи:

– processOpen (String message) – обробляє відкриття з’єднання із сервером;

– processClose (String message) – обробляє закриття з’єднання із сервером;

– processError (String message) – обробляє помилки при встановленні з’єднання;

– processMessage (String message) – обробляє повідомлення, що надходять від сервера;

– sendMessage (String code, String message) – формує JSON-пакет із кодом повідомлення code та текстом message і надсилає його серверу;

– processStates () – обробляє дії користувача залежно від поточного стану.

1. Клас Recognizer

Клас займається розпізнанням мовлення та перетворенням голосових запитів на текст.

Поля:

– langs – масив строк, у якому перераховано мови, на яких може вестись розпізнання;

– recognizing – показує, чи ведеться розпізнання в даний момент;

– recognition – об’єкт типу webkitSpeechRecognition, частина Google Speech API, відсилає запис фрази, сказаної користувачем, на сервер Google та отримує відповідь – ту ж фразу у вигляді тексту.

Методи:

– start (Event event) – розпочинає розпізнання;

– showInfo (String src, Element dest) – відображає інформацію src в елементі розмітки dest.

2.5 Діаграми послідовності

Діаграма послідовності (англ. sequence diagram) – діаграма, на якій показано взаємодію об’єктів (обмін між ними сигналами та повідомленнями), впорядковану за часом, з відображенням тривалості обробки і послідовності появи.

Основними елементами діаграми послідовності є:

– позначення об’єктів;

– вертикальні «лінії життя» (англ. lifeline), що відображають плин часу;

– прямокутники, що відображають діяльність об’єкта чи виконання ним певної функції;

– стрілки, що показують обмін сигналами чи повідомленнями між об’єктами.

Основні діаграми послідовності, розроблені для системи розпізнання голосових запитів для пошуку залізничних квитків та супутньої інформації, із коментарями розглянуто у подальших підрозділах.

1. Діаграма послідовності для процесу завантаження сторінки

Зазначену діаграму відображено на рис. 2.3.

Перш ніж сторінку буде завантажено, програма за допомогою геолокації встановить місцезнаходження користувача, визначить, чи є в базі даних місто, в якому знаходиться користувач, і якщо так, запропонує встановити це місто у якості пункту відправлення.

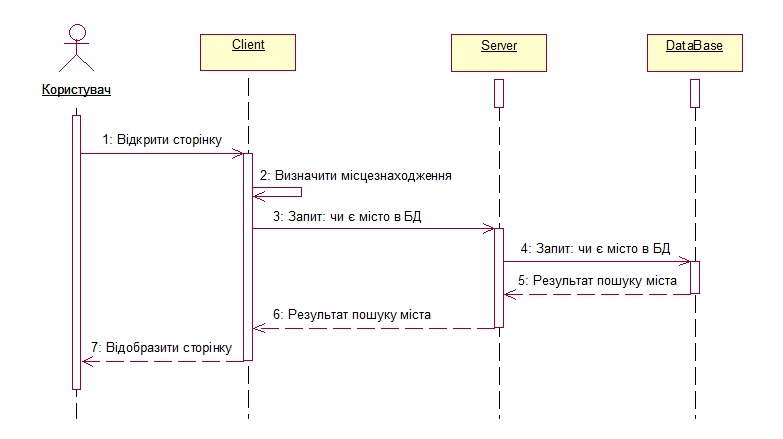


Рисунок 2.3 – Діаграма послідовності для процесу завантаження сторінки

1. Діаграма послідовності для етапів введення назв міст

Зазначену діаграму відображено на рис. 2.4.

Коли користувач вимовлятиме назви міст (для встановлення у якості пунктів призначення або прибуття), програма перевірятиме, чи є такі міста в базі даних. Якщо так, система зможе перейти до наступного етапу. Якщо такого міста у базі не виявиться, програма повідомить про це і попросить назвати інше місто.

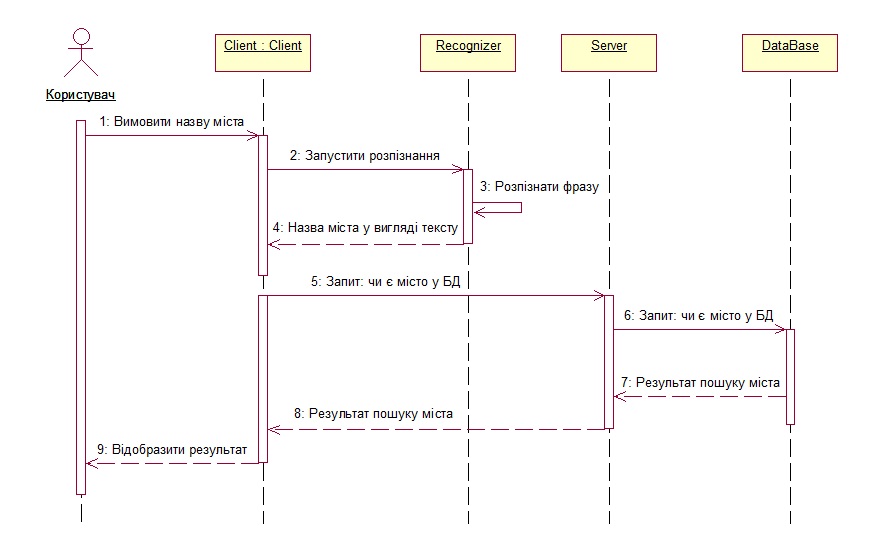


Рисунок 2.4 – Діаграма послідовності для етапів введення назв міст

1. Діаграма послідовності для етапу введення дати відправлення

Зазначену діаграму відображено на рис. 2.5.

Коли користувач вимовлятиме дату відправлення, програма перевірятиме, чи коректна ця дата (фраза користувача має містити число та місяць). Якщо користувач коректно вказав дату і ця дата настане не більше, ніж через місяць, система зможе перейти до наступного етапу. Якщо дату було вказано некоректно або вона настане більше, ніж через місяць, програма повідомить про помилку і попросить правильно указати дату.

1. Діаграма послідовності для етапу пошуку рейсів

Зазначену діаграму відображено на рис. 2.6.

Після того, як усі параметри буде введено, система розпочне пошук квитків, що задовольнятимуть указаним вимогам. Результатом буде список знайдених рейсів. Якщо жодного рейсу не буде знайдено, система запропонує вказати інші параметри.

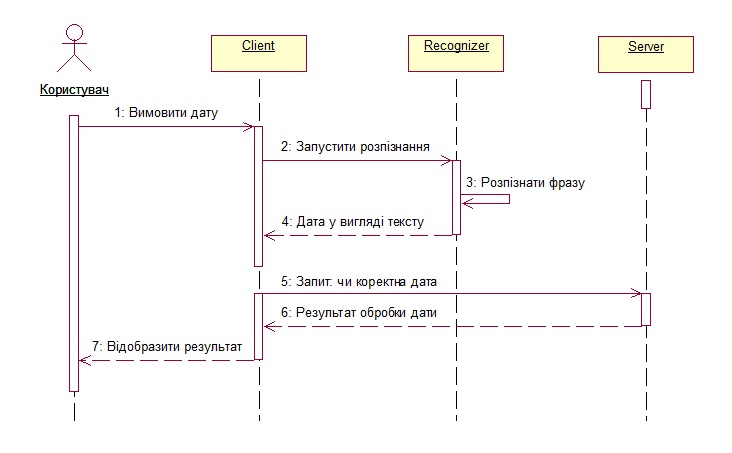


Рисунок 2.5 – Діаграма послідовності для етапу введення дати відправлення

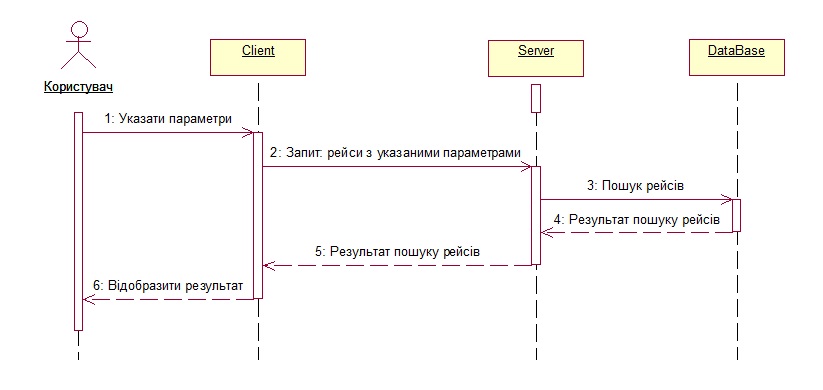


Рисунок 2.6 – Діаграма послідовності для етапу пошуку рейсів

1. Структура програми

Структуру програми наведено на рис. 2.7.

Користувач працює із веб-сторінкою index.htm, на якій відображається уся необхідна інформація. Стилі, що використовуються для оформлення цієї сторінки, зберігаються у таблиці стилів styles.css. Скрипти знаходяться у файлах scripts.js та recognition.js. Також використовуються зображення та аудіо файли. Клієнт сполучається із сервером за допомогою технології WebSocket. База даних обробляється у СУБД MySQL, до якої сервер звертається через драйвер JDBC.

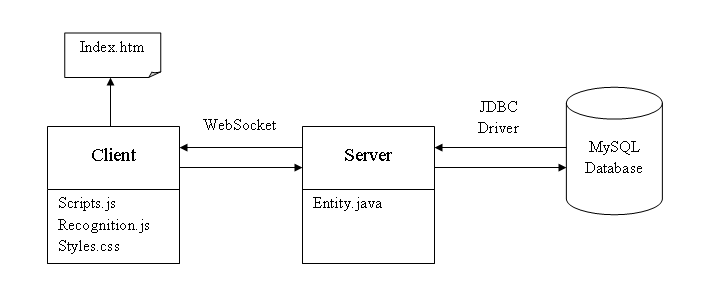


Рисунок 2.7 – Структура програми

3 РОЗРОБКА ПРОГРАМИ

1. Вибір мови програмування

Серверну частину програми написано з використанням мови Java. Такий вибір зроблено через такі особливості цієї мови:

* незалежність від операційної системи та обладнання;
* автоматичне управління пам’яттю;
* розширені можливості обробки виключних ситуацій;
* наявність стандартних колекцій (масив, список, стек і т.д.);
* наявність класів, що дозволяють виконувати HTTP-запити і обробляти відповіді;
* вбудовані засоби створення багатопоточних програм;
* уніфікований доступ до баз даних (на рівнях окремих SQL-запитів та концепції об’єктів, що володіють здатністю до збереження в базі даних);
* наявність зручних вільних середовищ розробки зі значним функціоналом (NetBeans, Eclipse, IntelliJ IDEA та ін.);
* наявність значної кількості сторонніх бібліотек, фреймворків (ORM, MVC та ін.), що суттєво розширює можливості та полегшує розробку.

Клієнтську частину програми написано з використанням мови JavaScript. Такий вибір зроблено через такі особливості цієї мови:

* підтримка найбільш популярними браузерами «за замовчуванням»;
* висока швидкість роботи;
* можливість роботи з DOM-моделлю веб-сторінки (як модифікувати сторінку, так і реагувати на події, які на ній відбуваються);
* скрипти не потребують компіляції, а підключаються до коду HTML і при завантаженні одразу виконуються;
* можливість виконувати асинхронні запити до сервера і отримувати відповідь без перезавантаження сторінки;
* наявність значної кількості сторонніх бібліотек та фреймворків.

Програмний засіб розроблено у середовищі NetBeans IDE 8.0.2. Такий вибір зроблено через наступні особливості цього засобу розробки:

* підтримка JDK 8 та Java EE 7;
* широкі можливості рефакторингу та профілювання;
* управління системами контролю версій;
* підтримка модульного тестування;
* встановлені фреймворки Spring MVC та Hibernate;
* встановлений драйвер JDBC;
* встановлений HTTP-сервер GlassFish.

1. Розробка інтерфейсу користувача

Інтерфейс користувача, або користувацький інтерфейс (англ. user interface, UI) – різновид інтерфейсів, в якому одна сторона представлена людиною (користувачем), а інша – машиною/пристроєм. Являє собою сукупність засобів і методів, за допомогою яких користувач взаємодіє із різноманітними машинами, пристроями, апаратурою.

Графічний інтерфейс користувача, або графічний користувацький інтерфейс (англ. graphical user interface, GUI) – різновид користувацького інтерфейсу, елементи якого (меню, кнопки, списки і т.п.), представлені користувачу на екрані, виконані у вигляді графічних зображень. На відміну від інтерфейсу командної строки, у графічному інтерфейсі користувач має довільний доступ до всіх видимих екранних об’єктів (елементів інтерфейсу) і здійснює безпосереднє маніпулювання ними. Графічний інтерфейс є частиною користувацького інтерфейсу і визначає взаємодію з користувачем на рівні візуалізованої інформації.

Інтелектуальний інтерфейс – такий різновид інтерфейсу, що збільшує діапазон способів введення та виведення інформації, збагачує граматики введення та виведення та намагається кооперуватися с користувачем для досягнення цілей. Містить модель предметної області для порівняння із контекстними умовами.

Інтерфейс розроблюваного програмного засобу є графічним та містить елементи інтелектуального (зокрема, голосове введення та виведення даних).

Розробка структури та сценарію діалогу

Діалог – це послідовність взаємодій між користувачем і програмою, в результаті яких розв’язуються певні задачі. Складність діалогу визначається функціональними можливостями додатка та можливими діями користувачів. Розвиток діалогу з часом можна розглядати як послідовність переходів системи з одного стану в інший. Жоден із цих станів не повинен бути «тупиковим», тобто користувач повинен мати можливість перейти з будь-якого поточного стану діалогу в необхідний (за один чи кілька кроків). Для цього необхідно визначити усі можливі стани діалогу та шляхи переходу з одного стану в інший.

Існують наступні варіанти структури діалогу:

* діалог типу «питання-відповідь»;
* діалог на основі меню;
* діалог на основі екранних форм;
* діалог на основі командної мови.

Кожний із цих варіантів має свої особливості і є найбільш зручним для певного класу задач. Для системи розпізнання голосових запитів для пошуку залізничних квитків та супутньої інформації було обрано структуру діалогу «питання-відповідь».

Структура діалогу типу «питання-відповідь» заснована на аналогії зі звичайним інтерв’ю. Система бере на себе роль опитувача та отримує інформацію від користувача у вигляді відповідей на питання. У кожній точці діалогу система виводить у якості підказки одне питання, на яке користувач дає одну відповідь. Залежно від отриманої відповіді система може вирішувати, яке питання поставити далі. Дана структура діалогу надає природний механізм введення як команд, так і даних. Особливо доцільна така структура в експертних системах, де діалог має багато розгалужень і для кожного питання передбачена велика кількість відповідей, кожна з яких впливає на те, яке питання буде поставлено наступним.

Розроблений сценарій діалогу відображено на рис. 3.1.

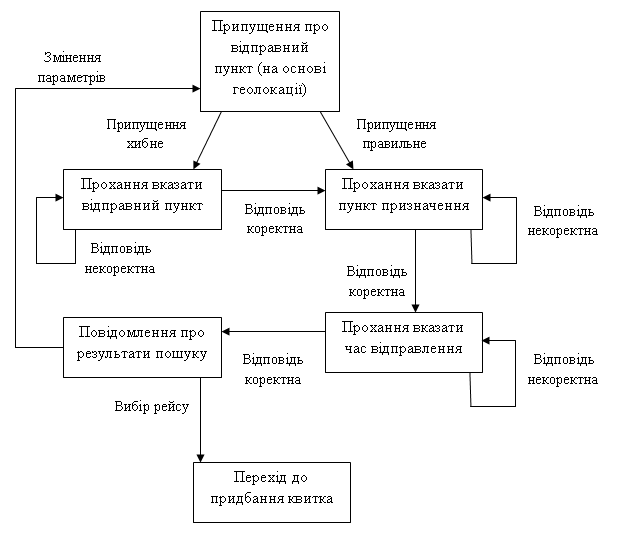


Рисунок 3.1 – Схема сценарію діалогу

Розробка інтерфейсу

На кожному з етапів роботи програми на екрані відображаються:

* текст запитання (складається із двох частин: безпосередньо запитання і його пояснення; програма також програє аудіо запис із текстом запитання);
* текст відповіді (текст, вимовлений користувачем; поки аналіз мовлення не завершено, цей текст буде сірого кольору, а коли завершено – чорного);
* підказки (щодо того, що треба зробити, щоб почати говорити, та що саме говорити).

Щоб почати говорити, треба натиснути і утримувати кнопку з піктограмою мікрофона на екрані або «space» (пробіл) на клавіатурі. При першому запуску програми необхідно дати їй дозвіл на використання мікрофона.

Приклади форматування екрану відображені на рисунках 3.2 – 3.4.

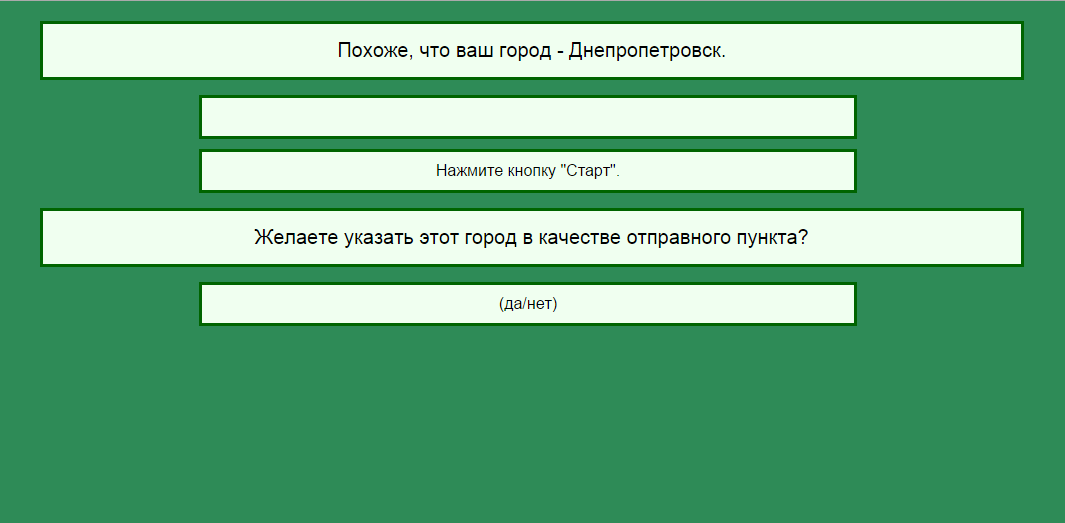


Рисунок 3.2 – Приклад форматування (етап підтвердження пункту відправлення)

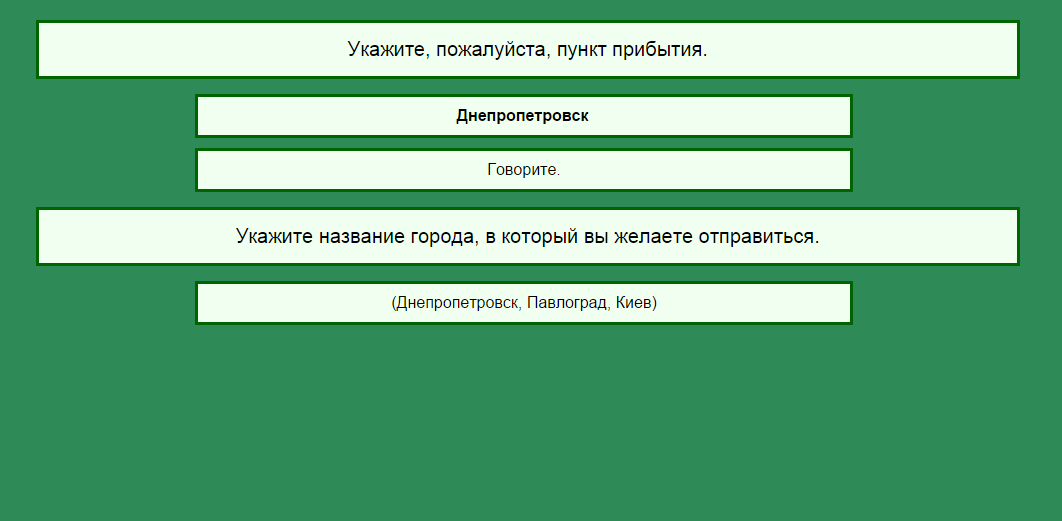


Рисунок 3.3 – Приклад форматування (етап визначення пункту прибуття)

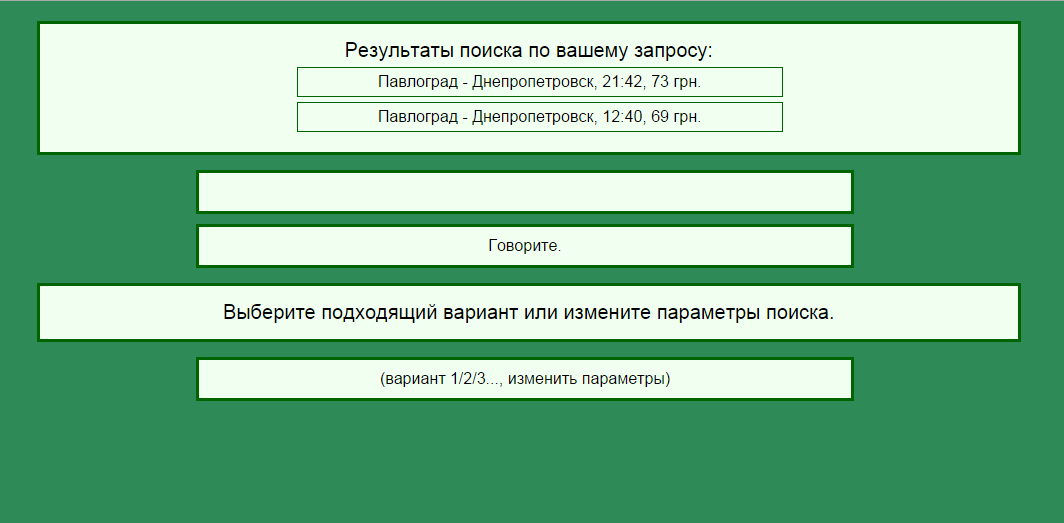


Рисунок 3.4 – Приклад форматування (етап вибору рейсу)

1. Розробка бази даних
2. Загальні відомості про бази даних

База даних – сукупність даних, організованих відповідно до концептуальної структури, що описує характеристики цих даних та відносини між ними. База даних є інформаційною моделлю певної предметної області.

Предметна область – частина реального світу, множина усіх предметів, властивості яких і відносини між якими розглядаються у межах даного контексту.

Відомі три різновиди структури даних: ієрархічна, мережева і таблична. Відповідно, за ознакою структури бази даних розподіляються на ієрархічні, мережеві і табличні (реляційні). Найбільш поширеним типом є реляційні бази даних. Відомо, що будь-яку структуру даних можна звести до табличної форми. Для розроблюваної системи також буде використовуватись реляційна база даних.

Крім цього, класифікація баз даних можлива за характером інформації (фактографічні та документальні БД) та за способом зберігання даних (централізовані та розподілені БД).

Основні поняття організації даних у реляційних БД:

* таблиця – об’єкт бази даних, який слугує для зберігання інформації про певну сутність;
* запис – рядок таблиці, що містить набір значень властивостей, розташованих у полях бази даних;
* поле – стовпець таблиці, що містить значення певної властивості;
* тип поля – визначається типом даних, які в ньому містяться (основні типи даних: лічильник, текстовий, числовий, дата/час, грошовий, логічний, об’єкт OLE, гіперпосилання);
* головний ключ таблиці – поле, значення якого є унікальними та дають змогу однозначно визначити запис у таблиці.

СУБД (система управління базами даних) – програмне забезпечення для роботи з базами даних.

1. Вибір СУБД

Більш конкретно, система управління базами даних (СУБД) – це сукупність програмних та лінгвістичних засобів загального чи спеціального призначення, що забезпечують управління створенням та використанням баз даних.

Основні характеристики СУБД:

* контроль за надлишковістю даних;
* несуперечливість даних;
* підтримка цілісності бази даних (коректність та несуперечливість);
* цілісність описується за допомогою обмежень;
* незалежність прикладних програм від даних;
* спільне використання даних;
* підвищений рівень безпеки.

Можливості СУБД:

* дозволяється створювати БД (здійснюється за допомогою мови визначення даних DDL ([Data Definition Language](https://uk.wikipedia.org/wiki/Data_Definition_Language" \o "Data Definition Language)));
* дозволяється додавання, оновлення, видалення та читання інформації з БД (за допомогою мови маніпулювання даними [DML](https://uk.wikipedia.org/wiki/DML), яку часто називають мовою запитів);
* можна надавати контрольований доступ до БД за допомогою:
* системи забезпечення захисту, яка запобігає несанкціонованому доступу до БД;
* системи керування паралельною роботою прикладних програм, яка контролює процеси спільного доступу до БД;
* система відновлення, яка дозволяє відновлювати БД до попереднього несуперечливого стану, що був порушений в результаті збою апаратного або програмного забезпечення.

У якості СУБД для системи розпізнання голосових запитів для пошуку залізничних квитків та супутньої інформації біло обрано MySQL Server 5.5. Такий вибір зроблено через наступні особливості цієї СУБД:

* MySQL є вільною системою управління базами даних;
* MySQL портовано на більшість сучасних платформ;
* MySQL має API для мови Java;
* висока швидкодія (завдяки внутрішньому механізму багатопотоковості) та невибагливість до ресурсів;
* безпека ­(завдяки системі із п’яти службових таблиць завжди можна описати, який користувач з якого домену з якою таблицею може працювати і які команди може здійснювати);
* надійність та стабільність пакету.

1. Проектування бази даних

Основні задачі проектування:

* забезпечення зберігання в БД усієї необхідної інформації;
* забезпечення можливості отримання даних за всіма необхідними запитами;
* скорочення надмірності та дублювання даних;
* забезпечення цілісності бази даних.

1. Концептуальне проектування

Концептуальне (інфологічне) проектування – побудова семантичної моделі найбільш високого рівня абстракції. Така модель створюється без орієнтації на якусь конкретну СУБД чи модель даних. Вид і зміст концептуальної моделі бази даних визначається обраним формальним апаратом (зазвичай це графічні нотації, подібні до ER-діаграм).

Найчастіше концептуальна модель бази даних включає:

* опис інформаційних об’єктів чи понять предметної області і зв’язків між ними;
* опис обмежень цілісності (вимог до допустимих значень даних та зв’язків між ними).

Виходячи з прийнятих при проектуванні (зокрема в розділі 2.5.1) рішень, звертання до бази даних відбуватиметься у двох випадках:

* щоб перевірити наявність певного міста;
* щоб знайти рейс із указаними параметрами (пункт відправлення, пункт прибуття, час відправлення, вартість квитка).

Таким чином, достатньо, щоб у базі були описані дві сутності: місто та рейс. Модель, що описує їх, зображено на рис. 3.5.

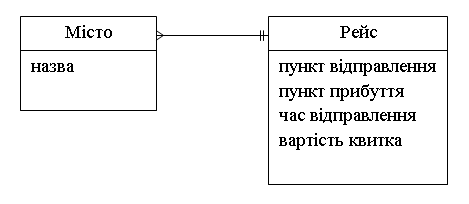


Рисунок 3.5 – Концептуальна модель бази даних

1. Логічне проектування

Логічне (даталогічне) проектування – створення схеми бази даних на основі конкретної моделі даних (реляційної, мережевої або ієрархічної).

Перетворення концептуальної моделі на логічну здійснюється за формальними правилами.

На етапі логічного проектування враховується специфіка конкретної моделі даних, але може не враховуватися специфіка конкретної СУБД.

Оскільки, як зазначалося у пункті 3.3.1, було прийнято рішення використовувати реляційну модель, дані у базі являтимуть собою набір відношень. База міститиме наступні таблиці:

* таблиця «Міста» (атрибути: ідентифікатор, назва);
* таблиця «Рейси» (атрибути: ідентифікатор, пункт відправлення, пункт прибуття, час відправлення, вартість квитка).

Оскільки пункти відправлення та прибуття є містами, у таблиці «Рейси» доцільно зберігати не їх назви, а відповідні ідентифікатори з таблиці «Міста». Ключовими в обох таблицях є поля «ідентифікатор».

Використання реляційної моделі передбачає проведення нормалізації відношень (перевірки на відповідність нормальним формам).

Нормальна форма – властивість відношення у реляційній моделі даних, що характеризує його з точки зору надмірності, яка потенційно призводить до логічно помилкових результатів вибірки чи зміни даних. Нормальна форма визначається як сукупність вимог, яким повинне задовольняти відношення.

Були розглянуті перша, друга та третя нормальні форми.

Перша нормальна форма утворює ґрунт для структурованої схеми баз даних:

* кожна таблиця повинна мати основний ключ: мінімальний набір колонок, які ідентифікують запис;
* уникнення повторень груп (категорії даних, що можуть зустрічатись різну кількість разів в різних записах) правильно визначаючи неключові атрибути;
* атомарність: кожен міста атрибут повинен мати лише одне значення, а не множину значень.

Спроектована модель відповідає вимогам першої нормальної форми.

Друга нормальна форма вимагає, аби дані, що зберігаються в таблицях із композитним ключем не залежали лише від частини ключа:

* схема бази даних повинна відповідати вимогам першої нормальної форми;
* дані, що повторно з'являються в декількох рядках виносяться в окремі таблиці.

Спроектована модель відповідає вимогам другої нормальної форми.

Третя нормальна форма вимагає, аби дані в таблиці залежали винятково від основного ключа:

* схема бази даних повинна відповідати всім вимогам другої нормальної форми;
* будь-яке поле, що залежить від основного ключа та від будь-якого іншого поля, має виноситись в окрему таблицю.

Спроектована модель відповідає вимогам третьої нормальної форми.

1. Фізичне проектування

Фізичне проектування – створення схеми бази даних для конкретної СУБД. Специфіка конкретної СУБД може включати в себе обмеження на іменування об’єктів бази даних, обмеження на типи даних, що підтримуються, і т.п. Крім того, специфіка конкретної СУБД при фізичному проектуванні може включати вибір рішень, пов’язаних із фізичним середовищем зберігання даних (вибір методів управління дисковою пам’яттю, розподіл бази даних за файлами та пристроями, методи доступу до даних), створення індексів і т.д.

Схема бази даних, розроблена для СУБД MySQL, відображена на рис. 3.6.

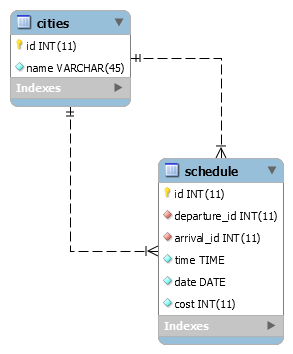


Рисунок 3.6 – Схема бази даних (результат фізичного проектування)

Таблиця «cities» описує міста. Таблиця містить наступні атрибути:

* ключове поле id;
* строкове поле name – назву міста.

Скрипт для створення таблиці на мові MySQL:

DROP TABLE IF EXISTS `cities`;

CREATE TABLE `cities` (

`id` int(11) NOT NULL AUTO\_INCREMENT,

`name` varchar(45) NOT NULL,

PRIMARY KEY (`id`))

ENGINE=InnoDB AUTO\_INCREMENT=28 DEFAULT CHARSET=cp1251;

Таблиця «schedule» описує рейси. Таблиця містить наступні атрибути:

* ключове поле id;
* поля departure\_id (пункт відправлення) та arrival\_id (пункт призначення), які пов’язані із ідентифікатором таблиці «cities»;
* поля time та date, що описують час відправлення (час і дата зберігаються окремо, оскільки використовуються також окремо);
* цілочислове поле cost (вартість квитка).

Скрипт для створення таблиці на мові MySQL:

DROP TABLE IF EXISTS `schedule`;

CREATE TABLE `schedule` (

`id` int(11) NOT NULL AUTO\_INCREMENT,

`departure\_id` int(11) NOT NULL,

`arrival\_id` int(11) NOT NULL,

`time` time NOT NULL,

`date` date NOT NULL,

`cost` int(11) NOT NULL,

PRIMARY KEY (`id`),

KEY `departure\_idx` (`departure\_id`),

KEY `arrival\_idx` (`arrival\_id`),

CONSTRAINT `arrival` FOREIGN KEY (`arrival\_id`) REFERENCES `cities` (`id`) ON DELETE RESTRICT ON UPDATE RESTRICT,

CONSTRAINT `departure` FOREIGN KEY (`departure\_id`) REFERENCES `cities` (`id`) ON DELETE RESTRICT ON UPDATE RESTRICT)

ENGINE=InnoDB AUTO\_INCREMENT=27379 DEFAULT CHARSET=cp1251;

1. Розробка алгоритму
2. Розробка алгоритму

4 ТЕСТУВАННЯ ТА ВІДЛАГОДЖЕННЯ