МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет кібербезпеки, комп'ютерної та програмної інженерії

Кафедра комп'ютерних інформаційних технологій

ДОПУСТИТИ ДО ЗАХИСТУ

Завідувач кафедри

Савченко А.С.

«___»_____2020 р.

ДИПЛОМНА РОБОТА

(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ВИПУСКНИКА ОСВІТНЬОГО СТУПЕНЯ

"МАГІСТР"

Тема: « Web-додаток "Сервіс електронних візиток" з клієнтом для ОС Android»

Виконав: Омельянюк Микита Олександрович

Керівник: к.т.н., професор Воронін Альберт Миколайович

Нормоконтролер з ЄСКД (ЄСПД): Райчев І.Е.

Київ 2020

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕСИТЕТ

Факультет кібербезпеки, комп’ютерної та програмної інженерії

Кафедра комп’ютерних інформаційних технологій

Спеціальність 122 “Комп’ютерні науки та інформаційні технології”

Спеціалізація «Інформаційні управляючі системи та технології (за галузями)»

ЗАТВЕРДЖУЮ

Завідувач кафедри

Савченко А.С.

“ ” _____2019р.

З А В Д А Н Н Я

на виконання дипломної роботи студента

Омельянюк Микити Олександровича

(прізвище, ім’я, по батькові)

1. Тема проекту (роботи): «Web-додаток "Сервіс електронних візиток" з клієнтом для ОС Android» затверджена наказом ректора №2175/ст. від 14.10.2019р..
2. Термін виконання проекту (роботи): з 14.10.2019р. по 09.02.2020р.
3. Вихідні данні до проекту (роботи): web-додаток, електронні візитівки.
4. Зміст пояснювальної записки (перелік питань, що підлягають розробці): вступ, аналітичний огляд і постановка завдання, висновок.
5. Перелік обов’язкового графічного матеріалу: графічний інтерфейс користувача web-додатку, інтерфейс користувача мобільного додатку.

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Етапи виконання дипломної роботи	Термін виконання етапів	Примітка
1.	Проаналізувати літературу та джерела за темою дипломного проекту.	13.10.19– 20.10.19	
2.	Розроблення та затвердження плану дипломного проекту.	21.10.19– 22.10.19	
3.	Провести консультації з науковим керівником щодо створення першого розділу.	23.10.19 – 27.10.19	
4.	Розробка розділу 1: Специфыка розробки web-додатку	30.10.19 – 11.11.19	
5.	Розробка розділу 2: Візитні картки, в їх теперішньому вигляді у сучасному суспільстві	12.11.19 – 22.11.19	
6.	Розробка розділу 3: Розробка web-додатку на мові програмування JAVA. Основні етапи розробки та проектування	23.11.19 – 02.12.19	
7.	Розробка розділу 4: Web-додаток «Сервіс електронних візиток» з клієнтом для ОС Android	03.12.19 – 11.12.19	
8.	Розробка розділу 5: Сервіси по розміщенню web-додатків та додатків під ОС Android	12.12.19 – 22.12.19	
9.	Висновки та оформлення пояснювальної записки дипломного проекту.	25.12.19 – 29.12.19	
10.	Підписання необхідних документів у встановленому порядку.	15.01.20-19.01.20	
11.	Підготовка до захисту та попередній захист дипломного проекту на випусковій кафедрі дипломного проекту	22.01.20 – 31.01.20	

7. Дата видачі завдання: 13.10.2019р.

Керівник дипломного проекту _____ Воронін А.М.
(підпис керівника) (П.І.Б.)

Завдання прийняв до виконання _____ Омельянюк М.О.
(підпис випускника) (П.І.Б.).

РЕФЕРАТ

Пояснювальна записка до дипломного проекту роботи « Web-додаток "Сервіс електронних візиток" з клієнтом для ОС Android» викладена на 86 с., містить 47 рис., 11 літературних джерел.

Ключові слова: WEB-ДОДАТОК, КЛІЄНТ, СЕРВЕР, МОБІЛЬНИЙ ДОДАТОК, ФРОНТЕНД, БЕКЕНД, JAVA, ANDROID.

Об'єкт дослідження: web-додаток з клієнтом під ОС Android, розроблений на мові програмування Java.

Предмет дослідження: альтернатива використання паперових візитних карток.

Мета роботи: розробка зручного додатку для використання візитних карток в електронному вигляді.

Методи дослідження: порівняльний аналіз отриманих даних для виявлення необхідності в викокористанні візитних карток в електронному ворматі.

Отримані результати: у процесі створення дипломної роботи були отримані певні резкльтати, які можна використовувати у майбутньому.

Результати дипломної роботи планується використовувати для подальшої розробки проектів з метою заміщення використання паперу та виробів з паперу, товарами в електронному вигляді.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ, ТЕРМІНІВ

JSON (JavaScript Object Notation) – текстовий формат обміну даними, заснований на JavaScript. Як і багато інших текстові формати, JSON легко читається людьми.

SDK (Software Development Kit) – певний набір засобів для розробки.

JDK (Java Development Kit) – набір засобів для розробки програм на мові програмування Java.

IDE (Integrated Development Kit) – інтегрована середа розробки.

JVM (Java Virtual Machine) – віртуальна машина Java.

GUI (Graphical User Interface) – графічний інтерфейс користувача.

UI (User Interface) – інтерфейс користувача.

ВСТУП

Сучасний світ тісно пов’язаний з IT. Люди все більше часу проводять в інтернеті, і вже звикли до того що основні банківські операції, покупки у магазині, отримання будь яких послуг та використання сервісів або ж оплату послуг та товарів можна робити через інтернет, не виходячи з дому. І для виконання цих дій існують безліч web-додатків. Але web-додатки у звичайному своєму уявленні не дуже зручні у користуванні, не зі стаціонарних пристроїв накшталт персонального комп’ютеру чи ноутбуку, а з портативних, таких як смартфон чи планшет, тому для облегшення цих задач у зв’язці з web-додатком розробляються мобільний додаток, який представляє собою звичайний додаток-клієнт, що не містить в собі ніякої складної бізнес логіки, а лише відображає інформацію і надає інтерфейс взаємодії з бекендом web-додатку, через API цього додатку.

Web-додаток – це клієнт-серверний додаток, у якому клієнт взаємодіє з сервером за допомогою браузера, а за сервер відповідає web-сервер. Логіка веб-додатку розподілена між сервером і клієнтом, при цьому зберігання даних здійснюється, переважно, на сервері. Обмін інформацією відбувається по мережі. Однією з переваг такого підходу є той факт, що клієнти не залежать від конкретної операційної системи користувача, тому веб-додатки є міжплатформенними службами.

При використанні мобільного додатку, як клієнту, взаємодія відбувається також по мережі але на відміну від web-клієнта (браузера), обмін інформацією забезпечується завдяки API, яке надає додаток і через яке реалізовано обмін інформацією, яка передається у форматі JSON а не передачею даних по HTTP.

Фронтендом є все те, що браузер може читати, виводити на екран та/або запускати, для мобільного додатку інтерфейс користувача або UI. На початку розвитку web-додатків, їх клієнтська частина уявляла собою майже статичну web-сторінку, де після виконання кожної дії (вибору певного

пункту або категорії, натискання кнопки), виконувалось повне перезавантаження усієї сторінки. Це є недуже економним рішенням в плані використання трафіку.

Зараз в основі UI лежить AJAX, що робить роботу з ними більш швидкою і економною в плані витрачання трафіку. Завдяки цьому, після виконання певних дій, web-сторінка оновлюється не цілком, а лише та частина де відбулися певні зміни. Фронтенд web-додатку реалізовується за допомогою мови розмітки HTML.

UI для додатку на платформі Android написано за допомогою розширюваної мови розмітки XML.

Для надання зовнішнього вигляду сторінок використовуються каскадні таблиці стилів – CSS, для надання якогось функціоналу на сторону клієнта або написання сценарію для web-сторінок і надання їм інтерактивності, використовують мову програмування JS.

РОЗДІЛ 1. Специфіка розробки web-додатку

Кожен web-додаток складається з клієнтської і серверної частин, тим самим реалізуючи технологію «клієнт-сервер».

Клієнтська частина реалізує користувальницький інтерфейс, формує запити до сервера і обробляє відповіді від нього.

Серверна частина отримує запит від клієнта, виконує обробку, після цього формує web-сторінку і відправляє її клієнту через мережу з використанням протоколу HTTP/HTTPS, але у випадку, якщо клієнтом виступає мобільний додаток, то на бекенді розробляється API, через яке відбувається взаємодія між серверною частиною додатку і мобільним пристроєм-клієнтом.

1.1. Загальні теоретичні відомості

HTTP та HTTPS протоколи

HTTP (англ. HyperText Transfer Protocol – «протокол передачі гіпертексту») – протокол прикладного рівня передачі даних (спочатку – у вигляді гіпертекстових документів в форматі «HTML», зараз використовується для передачі довільних даних). Основою HTTP є технологія «клієнт-сервер», тобто передбачається існування:

- Споживачів (клієнтів), які ініціюють з'єднання і надсилають запит;
- Постачальників (серверів), які очікують з'єднання для отримання запиту, виробляють необхідні дії і повертають назад повідомлення з результатом.

HTTPS (англ. HyperText Transfer Protocol Secure– «захищений протокол передачі гіпертексту») – розширення протоколу HTTP для підтримки шифрування з метою підвищення безпеки.

HTTP методи

За допомогою URL, ми визначаємо точну назву хоста, з яким хочемо спілкуватися, проте яку дію нам потрібно зробити, можна повідомити тільки за

допомогою HTTP методу. Звичайно ж існує кілька видів дій, які ми можемо зробити. У HTTP реалізовані найпотрібніші, які підходять під потреби більшості додатків.

Методи HTTP:

- GET: отримання доступу до існуючого ресурсу. В URL прихована вся необхідна інформація, щоб сервер міг знайти і повернути в якості відповіді запитуємий ресурс.
- POST: використовується для створення нового ресурсу. POST запит зазвичай містить в собі всю потрібну інформацію для створення нового ресурсу.

- PUT: оновити поточний ресурс. PUT запит містить оновлювані дані.
- DELETE: служить для видалення існуючого ресурсу.

HTTP методи, що обули представлені вище є найпопулярніші і найчастіше використовуються різними інструментами і фреймворкамі. У деяких випадках, PUT і DELETE запити відправляються за допомогою відправки POST, в змісті якого зазначено дію, яку потрібно зробити з ресурсом: створити, оновити або видалити.

Окрім методів: GET, POST, PUT і DELETE, HTTP підтримує і інші методи, такі як:

- HEAD: або ж аналогічний GET. Їх різниця в тому, що при даному виді запиту не передається повідомлення. Сервер отримує лише заголовки. Використовується, наприклад, для того щоб визначити, чи був змінений ресурс.
- TRACE: під час передачі запит проходить через безліч точок доступу і проксі серверів, кожен з яких вносить свою інформацію: IP, DNS. За допомогою даного методу, можна побачити всю проміжну інформацію.
- OPTIONS: використовується для визначення можливостей сервера, його параметрів і конфігурації для конкретного ресурсу.

URL

URL (адреса сайту або окремої сторінки) - це спеціальна форма позначення індивідуальної адреси ресурсу в інтернеті. В даному випадку під ресурсом можна розуміти сайт, окремий документ або зображення. Ввівши його URL в адресний рядок, користувач може відшукати його. Або ж URL-адреса (Uniform Resource Locator/Уніфікований Локатор Ресурсів), більш відомий як "веб-адреса", визначає розташування ресурсу (наприклад, веб-сторінки) в інтернеті. URL-адреса також визначає, як отримати цей ресурс, також відомий як "протокол", такий як HTTP, HTTPS, FTP тощо.

Шлях від сервера до кінцевого пристрою, з якого користувач виходить

на ресурс, можна проілюструвати нескладної схемою. Верхній елемент - це сервер ресурсу, а найнижчий - призначений для користувача девайс, всі проміжні точки - це додаткові сервери.

В даний час набирає популярність новий підхід до розробки веб-додатків, званий Ajax. При використанні Ajax сторінки web-додатку не перезавантажуються цілком, а лише довантажують необхідні дані з сервера, що робить їх більш інтерактивними і продуктивними.

API

API (Прикладниий програамний інтерфеейс англ. *Application Programming Interface*) — набір визначень підпрограм, протоколів взаємодії та засобів для створення програмного забезпечення. Спрощено - це набір чітко визначених методів для взаємодії різних компонентів. API надає розробнику засоби для швидкої розробки програмного забезпечення. API може бути для веб-базованих систем, операційних систем, баз даних, апаратного забезпечення, програмних бібліотек.

Сучасний API

Протягом багатьох років «API» часто описував будь-який тип інтерфейсу загального підключення до програми. Однак останнім часом сучасний API набув деяких характеристик, які роблять його більш цінними та корисними:

Сучасний API дотримується стандартів (як правило, HTTP та REST), які є зручними для розробників, легко доступними та зрозумілими.

До нього ставляться більше як до продуктів, ніж до коду. API розроблений для споживання конкретною аудиторією (наприклад, для розробників мобільних пдодатків), він документально описаний та розроблен таким чином, щоб користувачі могли мати певні

очікування щодо його використання, обслуговування та життєвого циклу.

Як і будь-яка інша частина виробленого програмного забезпечення, сучасний API має власний життєвий цикл розробки програмного забезпечення (software development lifecycle/SDLC), проектування, тестування, побудови, управління та контролю версій. Також сучасні API добре задокументовані для використання та контролю версій.

JSON

JSON (англ. JavaScript Object Notation, вимовляється як «джейсон») - текстовий формат обміну даними, заснований на JavaScript. Як і багато інших текстові формати, JSON легко читається людьми. Формат JSON був розроблений Дугласом Крокфорд .

Незважаючи на походження від JavaScript (точніше, від підмножини мови стандарту ECMA-262 1999 року), формат вважається незалежним від мови і може використовуватися практично з будь-якою мовою програмування. Для багатьох мов існує готовий код для створення і обробки даних у форматі JSON.

Бекендом web-додатку є все, що працює на сервері, тобто «не в браузері» або «на комп'ютері, підключеному до мережі (зазвичай до інтернету), який відповідає на повідомлення від інших комп'ютерів».

Для бекенд ви можете використовувати будь-які інструменти, доступні на вашому сервері (який, по суті, є просто комп'ютером, налаштованим для відповідей на повідомлення). Це означає, що ви можете використовувати будь-яку універсальну мову програмування: Ruby, PHP, Python, Java, JavaScript / Node, bash тощо. Це також означає, що ви можете використовувати системи управління базами даних, такі як MySQL, PostgreSQL, MongoDB, Cassandra, Redis, Memcached та інші.

Фронтенд web-додатку являє собою набір статичних і динамічних веб-

сторінок. Статична веб-сторінка - це сторінка, яка завжди відображається перед користувачем в незмінному вигляді. Web-сервер відправляє сторінку за запитом web-браузера без будь-яких змін. На противагу цьому, сервер вносить зміни в динамічну web-сторінку перед відправкою її браузеру. У зв'язку з тим що сторінка змінюється, вона називається динамічною.

1.2. Основні інструменти для розробки на мові програмування Java, підготовка до вибору середовища розробки

Одним з дуже вагомих критеріїв під час приготування до розробки будь –якого додатку є вибір середовища розробки, адже неправильний вибір, не зручного або ж не ефективного в плані швидкодії чи функціоналу середовища розробки може суттєво уповільнити цей процес. Хоча для написання java коду можна використовувати будь-який текстовий редактор, і збирати та компілювати проект через термінал (командну строку), для розробки потрібно всього лише встановити JDK і вказати шлях до неї, але краще всього використовувати яку небудь IDE.

JDK (Java Development Kit з англ. Набір розробника Java) - безкоштовно поширюваний компанією Oracle Corporation (раніше називалась Sun Microsystems) комплект розробника додатків на мові Java, що включає в себе компілятор Java (javac), стандартні бібліотеки класів Java, приклади, документацію, різні утиліти і виконавчу систему Java (JRE). До складу JDK не входить інтегроване середовище розробки на мові Java, тому розробник, що використовує тільки JDK, змушений використовувати сторонній текстовий редактор і компілювати свої програми, використовуючи утиліти командного рядка/терміналу.

JRE(The Java Runtime Environment з англ. Середовище виконання Java) - це набір програмних засобів для розробки програм на мовы програмування Java. Він поєднує в собі віртуальну машину Java (JVM), основні класи платформи та підтримуючі бібліотеки.

JRE є частиною Java Development Kit (JDK), але його можна

завантажити окремо. JRE спочатку був розроблений компанією Sun Microsystems Inc., дочірньою організацією корпорації Oracle.

JVM (A Java virtual machine з англ. Віртуальна машина Java) - це віртуальна машина, яка дозволяє комп'ютеру запускати програми Java, а також програми, написані іншими мовами, які також компілюються в байт-код Java. JVM деталізується специфікацією, яка формально описує, що потрібно для реалізації JVM. Наявність специфікації забезпечує сумісність програм Java у різних реалізаціях, так що авторам програм, що використовують комплект Java Development Kit (JDK), не потрібно турбуватися про ідіосинкразії базової апаратної платформи.

IDE (An integrated development environment з англ. Інтегроване середовище розробки) - це програмне забезпечення, яке надає комплексні засоби комп'ютерним програмістам для розробки програмного забезпечення. IDE зазвичай складається щонайменше з редактора вихідного коду, засобів автоматизації побудови та налагоджувача. Деякі IDE, такі як NetBeans та Eclipse, містять необхідний компілятор, інтерпретатор або обидва; інші, такі як SharpDevelop та Lazarus, не роблять.

Існує багато IDE для зручної та гнучкої розробки на мові Java, най популярніші з яких - це NetBeans, Eclipse, IntelliJ та IDEA . Тому при виборі середовища розробки, вибір зупинився між цима трьома IDE. Але для того щоб обрати конкретну, проведемо детальний аналіз, і потім зробимо вибір.

1.3. Вибір середовища для розробки web-додатку, та додатку під ОС Android

NetBeans IDE — вільне інтегроване середовище розробки (IDE) для мов програмування Java, JavaFX, C/C++, PHP, JavaScript, HTML5, Python, Groovy. Середовище може бути встановлене і для підтримки окремих мов, і у повній конфігурації. Середовище розробки NetBeans за замовчуванням підтримує розробку для платформ J2SE і J2EE.

Поширюється у сирцевих текстах під ліцензією Apache License. Проект NetBeans IDE підтримувався і спонсорувався фірмою Sun Microsystems і після придбання Sun — Oracle. У жовтні 2016 року Oracle передав NetBeans у власність Apache Software Foundation, яка займається розробкою і підтримкою проекту.

NetBeans IDE доступна для платформ Microsoft Windows, GNU/Linux, FreeBSD, і Solaris (як SPARC, так x86). Для інших платформ доступна можливість зібрати NetBeans самостійно із сирцевих текстів.

За якістю і можливостям останні версії NetBeans IDE змагається з найкращими інтегрованими середовищами розробки для мови Java, підтримуючи рефакторинг, профілювання, виділення синтаксичних конструкцій кольором, автодоповнення мовних конструкцій на льоту, шаблони коду та інше.

Eclipse IDE - є безкоштовною програмною платформою з відкритим вихідним кодом, контролюється організацією Eclipse Foundation. Написана на мові програмування Java і основною метою її створення є підвищення продуктивності процесу розробки програмного забезпечення.

Претендує на статус найбільш популярною Java IDE і є єдиним конкурентом такої потужної платформи як NetBeans.

Але на відміну від NetBeans який для створення елементів призначеного

для користувача інтерфейсу використовує від платформи незалежну бібліотеку Swing, в Eclipse використовується платформо-залежна бібліотека SWT - Standard Widget Toolkit.

IDE розроблені на базі платформи Eclipse застосовуються для створення програмного забезпечення на різних мовах програмування, так як Eclipse є платформою для розробки будь-яких інтегрованих середовищ програмування і розширень для себе ж, за принципом "Додатки для Eclipse розробляються в самій Eclipse".

Особливості платформи Eclipse:

- Кросплатформеність - працює під операційними системами Windows, Linux, Solaris і Mac OS X.

-Використовуючи Eclipse можна програмувати на багатьох мовах, таких як Java, C і C ++, PHP, Perl, Python, Cobol та інших.

-Є фреймворком для розробки інших інструментів і пропонує великий набір API для створення модулів.

-Використовуючи підхід RCP (Rich Client Platform) Eclipse є інструментом для створення практично будь-якого клієнтського програмного забезпечення.

InteliJ IDEA

IntelliJ IDEA — комерційне інтегроване середовище розробки для різних мов програмування (Java, Python, Scala, PHP та ін.) від компанії JetBrains. Система поставляється у вигляді урізаної по функціональності безкоштовної версії «Community Edition» і повнофункціональної комерційної версії «Ultimate Edition», для якої активні розробники відкритих проектів мають можливість отримати безкоштовну ліцензію. Сирцеві тексти Community-версії поширюються рамках ліцензії Apache 2.0. Бінарні збірки підготовлені для Linux, Mac OS X і Windows. А також є:

-Розумне автодоповнення, інструменти для аналізу якості коду, зручна навігація, розширені рефакторингом і форматування для Java, Groovy,

Scala, HTML, CSS, JavaScript, CoffeeScript, ActionScript, LESS, XML і багатьох інших мов.

-Підтримка всіх популярних фреймворків і платформ, включаючи Java EE, Spring Framework, Grails, Play Framework, GWT, Struts, Node.js, AngularJS, Android, Flex, AIR Mobile і багатьох інших.

-Інтеграція з серверами додатків, включаючи Tomcat, TomEE, GlassFish, JBoss, WebLogic, WebSphere, Geronimo, Resin, Jetty і Virgo.

-Інструменти для роботи з базами даних і SQL файлами, включаючи зручний клієнт і редактор для схеми бази даних.

-Інтеграція з комерційними системами управління версіями Perforce, Team Foundation Server, ClearCase, Visual SourceSafe.

-Інструменти для запуску тестів і аналізу покриття коду, включаючи підтримку всіх популярних фреймворків для тестування.

Зробивши детальний аналіз, вище вказаних IDE, було обрано IntelliJ IDEA, адже вона як найкраще задовольняла усім вимогам, а саме те що підтримувала багато мов програмування, адже окрім мови джава при написанні роботи використовувались ще: JavaScript, HTML та CSS.

Для написання ж додатку під OS Android, було обрано оду з най ефективніших і популярних IDE – AndroidStudio, від того ж самого виробника JetBrains, що і IntelliJ IDEA, яку було обрано для розробки web-додатку.

Android Studio — інтегроване середовище розробки (IDE) для платформи Android, представлене 16 травня 2013 року на конференції Google I/O менеджером по продукції корпорації Google — Еллі Паверс (англ. Ellie Powers). 8 грудня 2014 року компанія Google випустила перший стабільний реліз Android Studio 1.0.

Android Studio прийшло на зміну плаґіну ADT для платформи Eclipse. Середовище побудоване на базі вихідного коду продукту IntelliJ IDEA Community Edition, що розвивається компанією JetBrains.

Android Studio розвивається в рамках відкритої моделі розробки та поширюється під ліцензією Apache 2.0.

Не зважаючи на те що вище було обрано IntelliJ IDEA, як основний інструмент для розробки усього web-додатку, все ж для розробки окремо фронтенду, краще використовувати якийсь менш ресурсоємкий редактор коду, адже не завжди зручно для того щоб протестувати корректність відобрадення web-сторінки, «підіймати» увесь сервер. Таму було прийнято рішення обрати окремо редактор коду для розробки фронтенду web-додатку. На сьогоднішній день є велика кількість редакторів коду у яких можна займатися розробкою фронтенду, але кожен із них має свої позитивні і негативні сторони. Перед початком розробки було обрано два альтернативних варіанти редакторів коду, такі як: SublimeText3 і Atom. Детально проаналізувавши ці два редактори, вібір зупинився на SublimeText3.

1.4. Мови програмування і технології використані при розробці web-додатку та додатку під ОС Android

При розробці бекенду web-додатку та при розробці клієнта під OS Android, в переважній більшості використовувалвсь мова мова програмування Java, але іноді з’являлась необхідність у використанні JavaScript, для того щоб перенести частину навантаження (бізнеслогіки) з сервера на клієнта.

Java - строго типізований об'єктно-орієнтована мова програмування, розроблений компанією Sun Microsystems (в подальшому придбаної компанією Oracle). Розробка ведеться співтовариством, організованим через Java Community Process, мова і основні реалізують його технології поширюються за ліцензією GPL. Права на торговельну марку належать корпорації Oracle.

Програми Java зазвичай транслюються в спеціальний байт-код, тому вони можуть працювати на будь-якої комп'ютерної архітектурі за допомогою віртуальної Java-машини (JVM). Дата офіційного випуску - 23 травня 1995 року. На 2019 рік Java - одна з найпопулярніших мов програмування.

При розробці фронтенду використовувались такі мови програмування як HTML і JavaScript, бібліотека jQuery для більш легкої взаємодії JavaScript і HTML, каскадні таблиці стилів CSS, а для зменшення навантаження на сервер використовувалась технологія AJAX, завдяки чому тепер немає неохідності перезавантажувати усю сторінку цілком, щоб внести необхідні зміни на сторінці, а достатньо лише перезавантажити ту частину, до якої були внесені будь-які зміни і це значно зменшує навантаження на сервер.

Логотипи основних інструментів (мови гіпертекстової розмітки HTML, таблиці каскадних стилів CSS та мови програмування JS/JavaScript), що використовувались для розробки фронтенду web-додатку,

представлені на малюнку 1.1.

Мал.1.1. Логотипи основних інструментів що використовувались для розробки фронтенду web-додатку.

Розглянемо загально усі мови програмування і технології, що були задіяні:

HTML (HyperText Markup Language) – мова розмітки гіпертекстових документів, стандартна мова розмітки веб-сторінок в інтернеті.

JavaScript (JS) – динамічна, об'єктно-орієнтована прототипна мова програмування. Реалізація стандарту ECMAScript. Найчастіше використовується для створення сценаріїв веб-сторінок, що надає можливість на стороні клієнта (пристрої кінцевого користувача) взаємодіяти з користувачем, керувати браузером, асинхронно обмінюватися даними з сервером, змінювати структуру та зовнішній вигляд веб-сторінки.

CSS (Cascading Style Sheets, укр. Каскадні таблиці стилів) – спеціальна

мова, що використовується для опису зовнішнього вигляду сторінок, написаних мовами розмітки даних.

Найчастіше CSS використовують для візуальної презентації сторінок, написаних HTML та XHTML, але формат CSS може застосовуватися до інших видів XML-документів. Зокрема при розробці UI для додатку на ОС Android, для надання йому певного вигляду та стилю також було використано CSS.

jQuery – бібліотека JavaScript, що фокусується на взаємодії JavaScript і HTML. Бібліотека jQuery допомагає легко отримувати доступ до будь-якого елементу DOM, звертатися до атрибутів і вмісту елементів DOM, маніпулювати ними. Також бібліотека jQuery надає зручний API для роботи з AJAX.

AJAX

AJAX (Asynchronous Javascript and XML – «асинхронний JavaScript і XML») – підхід до побудови призначених для користувача інтерфейсів веб-додатків, що полягає в «фоновому» обміні даними браузера з веб-сервером. В результаті, при оновленні даних веб-сторінка не перезавантажується повністю, і веб-додатки стають швидше і зручніше.

SPRING FRAMEWORK

Spring Framework (або коротко Spring) - універсальний фреймворк з відкритим вихідним кодом для Java-платформи (мал.1.2.). Також існує форк для платформи .NET Framework, названий Spring.NET.

Мал.1.2. Офіційний логотип фреймворку Spring.

Перша версія була написана Родом Джонсоном, який вперше опублікував її разом з виданням своєї книги «Expert One-on-One Java EE Design and Development у жовтні 2002 року.

Фреймворк був вперше випущений під ліцензією Apache 2.0 license в червні 2003 року. Перша стабільна версія 1.0 була випущена в березні 2004. Spring 2.0 був випущений в жовтні 2006, Spring 2.5 - в листопаді 2007, Spring 3.0 в грудні 2009, і Spring 3.1 в грудні 2011. Поточна версія - 5.1.2.

Незважаючи на те, що Spring не забезпечував якусь певну модель програмування, він став широко розповсюдженим в Java-спільноті, головним чином як альтернатива і заміна моделі Enterprise JavaBeans. Spring надає велику свободу Java-розробникам в проектуванні; крім того, він надає добре документовані і легкі у використанні засоби вирішення проблем, що виникають при створенні додатків корпоративного масштабу.

Тим часом, особливості ядра Spring застосовні в будь-якому Java-додатку, і існує безліч розширень і удосконалень для побудови веб-

додатків на Java Enterprise платформі. З цих причин Spring набув великої популярності і визнається розробниками як стратегічно важливий фреймворк.

Spring забезпечує вирішення багатьох завдань, з якими стикаються Java-розробники і організації, які хочуть створити інформаційну систему, засновану на платформі Java. Через широку функціональності важко визначити найбільш значущі структурні елементи, з яких він складається. Spring не повністю пов'язаний з платформою Java Enterprise, незважаючи на його масштабну інтеграцію з нею, що є важливою причиною його популярності.

Spring, ймовірно, найбільш відомий як джерело розширень (features), потрібних для ефективної розробки складних бізнес-додатків поза великовагових програмних моделей, які історично були домінуючими в промисловості. Ще одне його достоїнство в тому, що він ввів раніше невикористовувані функціональні можливості в сьогоднішні панівні методи розробки, навіть поза платформи Java.

Цей фреймворк пропонує послідовну модель і робить її придатною до більшості типів додатків, які вже створені на основі платформи Java. Вважається, що Spring реалізує модель розробки, засновану на кращих стандартах індустрії, і робить її доступною в багатьох областях Java.

MVC

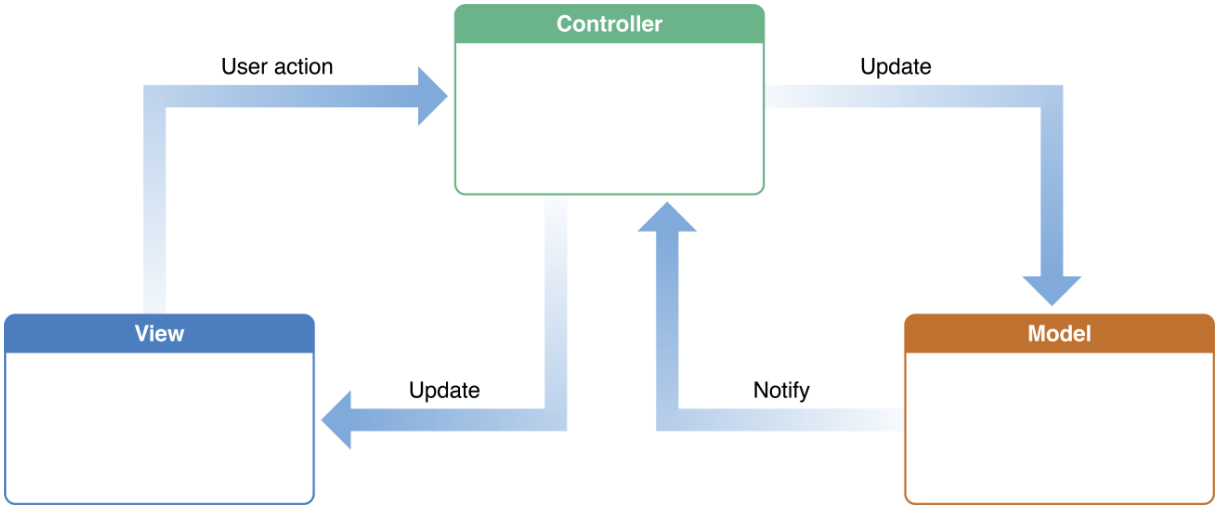
Model-View-Controller (MVC, «Модель-Представлення-Контролер», «Модель-Вид-Контролер») - схема поділу даних програми, призначеного для користувача інтерфейсу і керуючої логіки на три окремих компоненти: модель, уявлення і контролер - таким чином, що модифікація кожного компонента може здійснюватися незалежно.

Модель (Model) надає дані і реагує на команди контролера, змінюючи свій стан.

Подання (View) відповідає за відображення даних моделі користувачеві, реагуючи на зміни моделі.

Контролер (Controller) інтерпретує дії користувача, сповіщаючи модель про необхідність змін.

Схематично принцип роботи концепції MVC моказано на малюнку 1.3



Мал. 1.3. Схематичне зображення принципу роботи моделі MVC.

Концепція MVC була описана Трюгве Реенскаугом в 1978 році, який працював в науково-дослідному центрі «Xerox PARC» над мовою програмування «Smalltalk». Пізніше, Стів Бурбек реалізував шаблон в Smalltalk-80 .

Остаточна версія концепції MVC була опублікована тільки в 1988 році в журналі «Technology Object».

Згодом шаблон проектування став еволюціонувати. Наприклад, була представлена ієрархічна версія HMVC; MVA , MVVM.

Подальший виток популярності привнесло розвиток фреймворків, орієнтованих на швидку розгортку, на мовах Python і Ruby, Django і Rails. На момент 2017 року, фреймворки з MVC зайняли передові позиції що до решти фреймворків що не використовували даний шаблон.

Одією з основних, та дуже значних частин будь якого додатку є необхідність зберігання інформації та даних, з якими будуть проводитись операції, тому для цього необхідно обрати методи зберігання цієї інформації та даних. Най зручнішим способом зберігання великої кількості данних є використання баз данних.

База даних (БД) — це організована структура, яка призначена для зберігання, зміни та обробки взаємозалежної інформації, в переважній більшості великих обсягів. База даних — це певний набір даних, які пов'язані між собою спільною ознакою або властивістю і впорядковані, за якимось критерієм, наприклад, за алфавітом.

Об'єднання великої кількості даних в єдину базу дає змогу для формування безлічі варіації групування інформації — особисті дані клієнта, історія замовлень, каталог товарів та будь-що інше.

Головною перевагою БД є швидкість внесення та використання потрібної інформації. Завдяки спеціальним алгоритмам, які використовуються для баз даних, можна легко знаходити необхідні дані всього за декілька секунд. Також в базі даних існує певний взаємозв'язок інформації: зміна в одному рядку може спричинити зміни в інших рядках — це допомагає працювати з інформацією простіше і швидше.

Бази даних для web-додатків дають змогу зберігати інформацію, що виглядає як зв'язані між собою таблиці. Саме в БД зберігаються вся

необхідна та корисна інформація для функціонування сайту (клієнтські дані, прайс-лист, список товарів).

Щоб створити запит до бази даних часто використовують Structured Query Language. SQL дає змогу додавати, редагувати та видаляти інформацію, що міститься у таблицях.

Під час розробки різних web-додатків використовують різні системи управління БД. До основних СУБД, відносять:

- об'єктно-реляційна система управління базами даних Oracle Database;
- вільна система управління базами даних PostgreSQL;
- система керування базами даних Microsoft SQL Сервер;
- вільна система управління базами даних MySQL;

Такі системи управління відрізняються централізованою обробкою запитів, забезпечують надійність, доступність та безпеку БД.

При розробці web-додатку Сервіс електронних візиток" з клієнтом для ОС Android було обрано СКБД PostgreSQL.

PostgreSQL — об'єктно-реляційна система керування базами даних (СКБД). Є альтернативою як комерційним СКБД (Oracle Database, Microsoft SQL Server, IBM DB2 та інші), так і СКБД з відкритим кодом (MySQL, Firebird, SQLite)(мал.1.4.).

Мал.1.4. Логотип PostgreSQL.

Порівняно з іншими проектами з відкритим кодом, такими як Apache, FreeBSD або MySQL, PostgreSQL не контролюється якоюсь однією компанією, її розробка можлива завдяки співпраці багатьох людей та компаній, які хочуть використовувати цю СКБД та впроваджувати у неї найновіші досягнення.

Сервер PostgreSQL написаний на мові C. Зазвичай розповсюджується у вигляді набору текстових файлів із сирцевим кодом. Для інсталяції необхідно відкомпілювати файли на своєму комп'ютері і скопіювати в деякий каталог. Весь процес детально описаний в документації.

Як вже зазначалося вищє, для зберігання даних мжна використовувати різні бази даних - Oracle, MS SQL Server, MySQL, Postgres і т.д. Всі ці системи упраління базами даних мають свої особливості. Головне, що їх об'єднує це взаємодія зі сховищем даних за допомогою команд SQL. І щоб визначити єдиний механізм взаємодії з

цими СУБД в Java ще починаючи з 1996 року було введено спеціальний прикладний інтерфейс API, який називається JDBC.

Тобто якщо ми хочемо в додатку на мові Java взаємодіяти з базою даних, то необхідно використовувати функціональні можливості JDBC. Даний API входить до складу Java (на поточний момент це версія JDBC 4.3), зокрема, для роботи з JDBC в програмі Java досить підключити пакет java.sql. Для роботи в Java EE є аналогічний пакет javax.sql, який розширює можливості JDBC.

Для зручної та швидкої роботи додатку з БД при розробці на Java часто використовуэться JPA (Java Persistence API) –це специфікація API для Java EE, що надає можливість в базі даних Java об’єкти в зручному, для роботи, вигляді.

Однією з най популярніших і зручних JPA для роботи з БД на Java є Hibernate.

Hibernate

Hibernate - найпопулярніша реалізація специфікації JPA, призначена для вирішення завдань об'єктно-реляційного відображення (ORM). Поширюється вільно на умовах GNU Lesser General Public License (мал.1.5.).

Мал. 1.5. Логотип Hibernate.

Метою Hibernate є звільнення розробника від значного обсягу порівняно низкорівневого програмування при роботі в об'єктно-орієнтованих засобах в реляційної базі даних. Розробник може використовувати Hibernate як в процесі проектування системи класів і таблиць «з нуля», так і для роботи з уже існуючою базою даних.

Бібліотека не тільки виконує задачу зв'язку Java-класів з таблицями бази даних (і типів даних Java з типами даних SQL), але і також надає кошти для автоматичної генерації і оновлення набору таблиць, побудови запитів і обробки отриманих даних і може значно зменшити час розробки, яке зазвичай витрачається на ручне написання SQL- і JDBC-коду. Hibernate автоматизує генерацію SQL-запитів і звільняє розробника від ручної обробки результуючого набору даних і перетворення об'єктів, максимально полегшуючи перенесення (портирование) додатки на будь-які бази даних SQL.

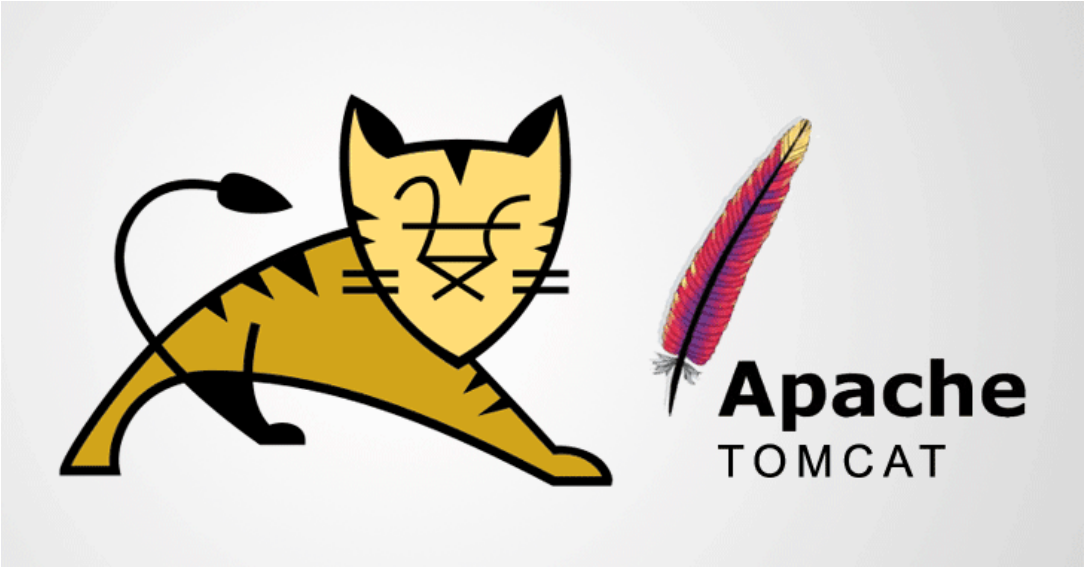
Mapping (зіставлення, проектування) Java-класів з таблицями бази даних здійснюється за допомогою конфігураційних XML-файлів або Java-анотацій. При використанні файлу XML Hibernate може генерувати скелет вихідного коду для класів тривалого зберігання. У цьому немає необхідності, якщо використовується анотація. Hibernate може використовувати файл XML або анотації для підтримки схеми бази даних.

Забезпечуються можливості по організації відносини між класами «один-до-багатьох» і «багато-до-багатьох». На додаток до управління зв'язками між об'єктами Hibernate також може управляти рефлексивними відносинами, де об'єкт має зв'язок «один-до-багатьох» з іншими екземплярами свого власного типу даних. Hibernate підтримує відображення для користувача типів значень. Це робить можливими такі сценарії:

- Перевизначення типу за замовчуванням SQL, Hibernate вибирає при відображенні стовпчика властивості.
- Проектування перераховується типу Java на поле БД, ніби вони є звичайними властивостями.
- Проектування одієї властивості в декілька колонок.

Ще одним дуже вагомим кроком є обрання серверу, на якому буде «підійматися» та працювати web-додатк. Таким сервером будо обрано Tomcat.

Програмне забезпечення Apache Tomcat (мал.1.6.) - це реалізація з відкритим кодом технологій Java Servlet, JavaServer Pages, Java Expression Language та Java WebSocket. Java сервлет, сторінки JavaServer, мова виразів Java та специфікації Java WebSocket розробляються в рамках процесу спільноти Java.



Мал.1.6. Офіційний логотип Apache Tomcat.

Програмне забезпечення Apache Tomcat розробляється у відкритому середовищі та бере участь у спільноті та випускається під ліцензією Apache License 2. Проект Apache Tomcat призначений для співпраці

кращих розробників з усього світу. Запрошуємо Вас взяти участь у цьому відкритому проекті розвитку. Щоб дізнатися більше про залучення, натисніть тут.

Програмне забезпечення Apache Tomcat використовує численні масштабні веб-програми, що мають найважливіше значення для місії, у різних галузях та організаціях.

ВИСНОВОК ДО РОЗДІЛУ 1

В першому розділі було розглянуто особливості розробки web-додатку на мові програмування Java, та специфіку розробки додатку-клієнту, що працює під керуванням ОС Android і розробки для нього API, для сумісної та зручної роботи мобільного додатку з web-додатком.

Головною перевагою web-додатку над десктопними додатками (тими що необхідно встановлювати на пристрій) є те що web-додатком можна буде користуватися в незалежності від платформи (операційної системи), а внаслідок цього немає необхідності вест розробку під декілька платформ, що значно скорочує час і фінансові витрати на розробку web-додатку, на відміну від десктопних додатків, які необхідно розробляти по різному під різні платформи.

Отже, можна зробити висновок, що у сучасному суспільстві дійсно є необхідність розробки додаткового ПО, окремо від web-версій, для більш гнучкої та зручної роботи з певними ресурсами та додатками.

РОЗДІЛ 2. Візитні картки, в їх теперішньому вигляді у сучасному суспільстві

Візитна картка (візитка) - загальноприйнятий носій контактної інформації про людину або організацію. Виготовляється з різних матеріалів: паперу, картону (мал.2.1.) або пластику невеликого формату, існує також варіант CD-візитки, виконаний на зменшеній до 50 мм на 90 мм поверхні CD-диска. Існують також візитні картки, виготовлені з дерева (дерев'яний шпон або фанера 3 мм завтовшки) і металу. Візитка включає ім'я власника, компанію, логотип і контактну інформацію (адреса, телефонний номер та / або адреса електронної пошти).

Мал.2.1. Приклад візитної картки виробленої з картону.

У переносному сенсі вираз «візитна картка чого-небудь» означає будь-яку відмітну, дуже характерну (як правило, позитивну) ознаку, що однозначно вказує на її володаря, і, як правило, на те що принесло йому широку популярність.

2.1. Види візиток, їх призначення та історія розвитку

Візитки умовно можна розділити на три види: особисті (сімейні), ділові та корпоративні.

Особисті візитки в основному використовуються в неформальному спілкуванні при дружньому знайомстві. Також популярні вони серед фрілансерів. У такій візитці, як правило, вказуються ім'я, прізвище та номер телефону власника. Посада і адреса в цій візитці можна опустити. Друк візитки може бути виконана в будь-якому стилі і розроблятися відповідно до індивідуальних переваг власника.

Корпоративна візитна картка, як правило, не містить імен і прізвищ. При друку в ній вказується інформація про компанію, сфера діяльності, перелік послуг, що надаються, контактні телефони, карта проїзду, адреса веб-сторінки. Зазвичай корпоративна візитка відображає фірмовий стиль компанії. Має рекламний характер і в основному використовується на виставках, конференціях, з'їздах.

Ділова візитка використовується в бізнесі, на офіційних зустрічах і переговорах, для надання контактної інформації своїм майбутнім клієнтам. На ділових візитках обов'язково вказуються ім'я, прізвище, посада бізнесмена, а також назва фірми і вид її діяльності. Відповідно до загальносвітової практики спочатку повинні бути вказані ім'я, по батькові, а потім - прізвище. У розробці візитки використовується фірмовий стиль компанії і логотип. Такі візитки зазвичай мають строгий дизайн. У державних службовців та депутатів на візитній картці може знаходитися зображення державних відзнак, таких як прапор і герб країни. Ділова візитна картка без адреси також не відповідає нормам етикету. Винятком є дипломати і вищі державні посадові особи. Ділові візитки повинні містити максимально читаються шрифти. Не рекомендується застосовувати складні декоративні шрифти (якщо тільки цього не вимагає профіль спектр ваших послуг), курсивне, а також жирне накреслення.

Згідно з першими згадками в історії, візитки з'явилися в Стародавньому Китаї, між другим і третім століттям до нашої ери. Китайські чиновники спеціальним указом зобов'язувалися мати картки на червоному папері з написаними на них ім'ям і посадою. Ці візитки можуть і зараз служити зразком стриманості і естетичності: ніяких зайвих і недоречних деталей, окрім імені, прізвища та посади.

Зразки ранніх китайських візиток епохи Західна Хань - були виявлені в Хуаншіянь в провінції Цзянсу (мал. 2.2.). Вони являють собою пластинки з дерева 21,5 × 6,5 см. Крім вручення при візитах, вони також використовувалися в похованнях для вказівки імені та статусу померлого.

Мал.2.2. Приклад однієї з перших візитних карток, знайденої у Кикаї.

Перша офіційно-історична згадка про візитці відноситься до часів правління короля Людовика XIV. З'явившись у Франції, візитка стала необхідністю, знаковим атрибутом для представників вищих верств населення того часу, ставши для них обов'язковим аксесуаром. Маючи вид грального картки з ім'ям візитера (звідки і пішла назва візитки – з фр. Visite - візит), візитка презентувала свого володаря з кращих його сторін.

Перша надрукована візитка була знайдена в Німеччині і датується 1786 роком. Візитки ставали необхідністю з цілим переліком правил

етикету за їх зверненням. Німецька аристократія слідувала французькому досвіду. Згодом візитки стали атрибутом дам і панів середнього класу і були відсутні у простих станів.

У XVI-XVII століттях гравірування візитки (тоді вони називалися - «візит-білетте») вже мали городяни Флоренції і Венеції. Саме там в той час була добре розвинена поліграфія. Виготовлення візиток набуло обрисів особливого виду мистецтва, займалися яким одні з кращих майстрів того часу.

Фотографи-портретисти, щоб успішно конкурувати з літографіями і граверами, прагнули робити фотографії великого розміру і більш вражаючі (щоб затримували на собі увагу). У 1854 році французький фотограф Андре-Адольф-Ежен Дісдері запатентував в Парижі «карт де візит» - фотоапарат з чотирма об'єктивами, яка робила вісім невеликих фотографій розміром 3,25 на 1,125 дюйма на «повної» фотоплатівці формату 6,5 на 8,5 дюйма. Ці вісі фотографії, кожна з яких представляла собою візитну картку розміром 4 на 2,5 дюйма, продавалися приблизно за 4 долари, більш ніж удвічі дешевше того, що звичайно запитували портретні фотографи за один повнорозмірний відбиток. Нововведення набуло несподіваний успіх, дозволивши не тільки розширити коло потенційних клієнтів, але і стати предметом колекціонування. Величезним попитом стали користуватися візитки знаменитостей, які поклали край початок культу зірок. Але в 1866 році ненаситний попит на «карт де візит» так само несподівано припинився, як і почався. Дісдері вигадував різні нововведення, щоб повернути до життя свій згасаючий бізнес, або насаджував трюкацтво на вже існуючі фотографії на кшталт фотографії на шовку, на кераміці. Ніщо не допомагало. Він не зміг перемогти ту дешевизну, яку сам породив, - ціна «карт де візит» впала до одного долара за дюжину (за 12 штук).

Потрібно було знайти спосіб якось зберігати тисячі карток, отримані від родичів або друзів, які залишали їх після візитів або обмінювалися

ними в дні народження і свят. Вихід був знайдений у вигляді альбомів для візитних карток. Деякі продавалися за звичайною ціною, але були і майстерно зроблені, переплетені добре обробленої і дорогою шкірою. Такий альбом став обов'язковою приналежністю, незамінним предметом бесіди в кожному салоні і художньої майстерні того часу.

2.2. Прблема візитних крток у їх теперішньому вигляді та методи їх рішенн

На теперішній час існує декілька проблем, пов’язаних з використанням візиток у тому вигляді якому ми їх маємо, і дві най важливіші проблеми це: те, як їх зберігати у такій кількості що ми маємо, адже їх може назбиратись велика кількість за дуже короткий період часу, та ще одна, але навіть більш значна в своєму сенсі – це те, скільки треба витратити деревини на виробництво певної кількості візитних карток, до того відсотку цих карток, які дійсно знадобляться і не будуть викинуті, за кілька днів або ж навіть хвилин після їх отримання, у сміття. За різними даними, на виробництво однієї тони паперу йде від 3,5 до 5 тон деревини. І не зважаючи на те що деревина вважається відновлювальним природним ресурсом, все одно це не означає шо треба вирубувать величезні площі лісів за для виробництва тих самих візитних карток, адже за статистикою дві третини усїх візитних карток, що були тільки передані людині до рук одразу ж іде до смітника, а лишившася третина з 75%-ою вірогідністю буде лежати без діла у гаманці, кармані або у ті самій візитниці і ніколи не буде використана, або просто буде загублена чи викинута через нетривалий проміжок часу.

Та навіть це ще не все, адже лищається ще та кількість візитних карток, що людина порахувала за потрібні собі, і вирішила лишити, а тут трапляється проблем, не дотримання людиною що виготовляла/замовляла виготовення даної візитки певних стандартів та норм виготовлення візитки і більшість візитних карток різного розміру та формату, і їх не зручно носити гаманці/візитниці, через одне просте «але», вона просто туди не вміщується (мал.2.3.).

Мал.2.3. Приклад візиток різних форматів.

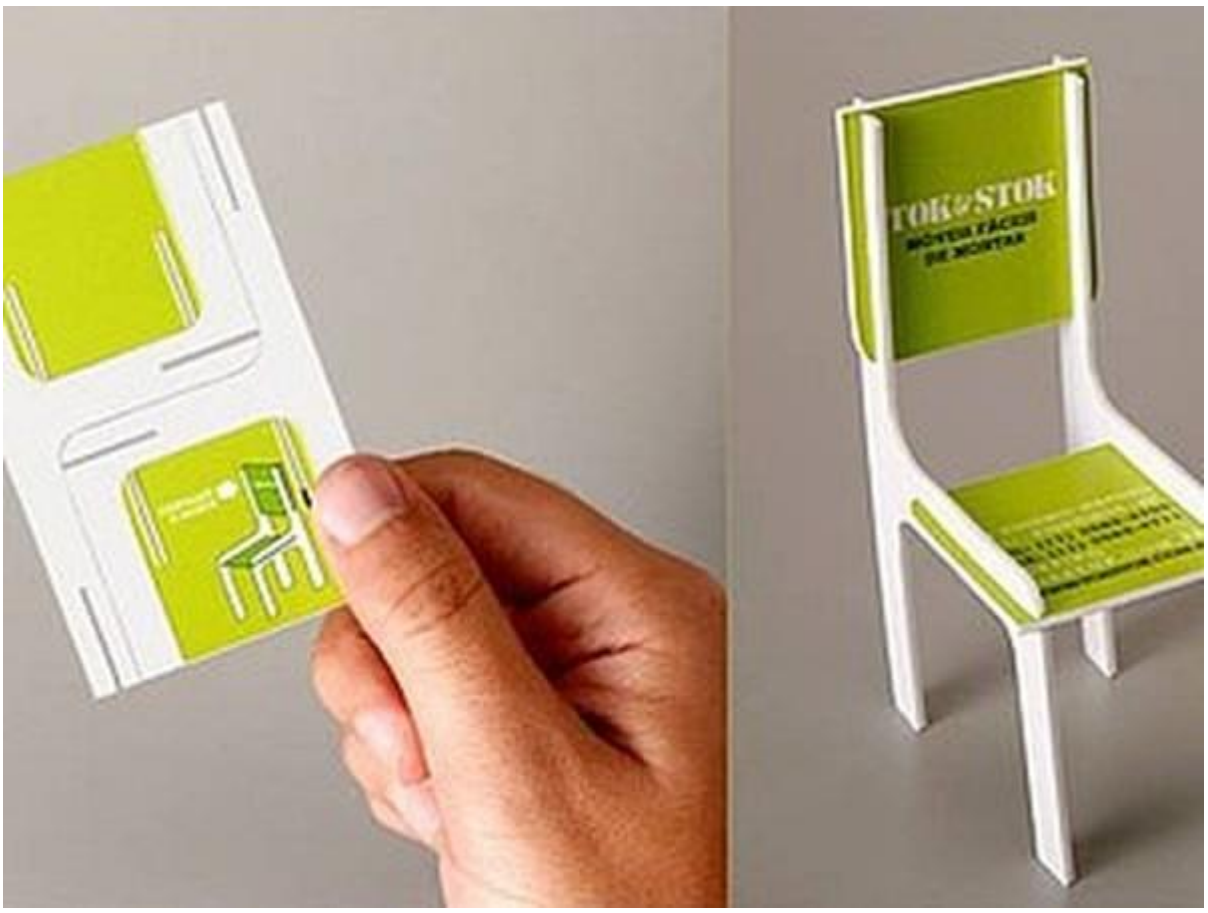
У наш час аби візитна картка одразу ж піся її отримання не загубилася серед десятків, а інколи і сотень інших візиток що людина має у себе, замовники (тримачі) тих самих карток прибігають до дуже креативних кроків при їх виготовленні, одним із них є не стандартний підхід до дизайну, такий приклад можна побачити на нижче на малюнках, де показано візитку фотографа-оператора, виконану на прозолому шматку пластику, у вигляді об’єктиву фотокамери (мал.2.4.),

Мал.2.4. Візитна картка фотографа-оператора.

візитну картку, виконану на замовлення ландшафтного дизайнера, у вигляді частини галявини, з дизайнерською підстрижкою газону, що на додачу ще скручується в рулон (мал.2.5.)

Мал.2.5. Візитна картка ландшафтного дизайнера.

та візитку майстра з виготовлення меблв, яка на перший погляд схожа на візитну картку стандартних розмірів і формату, але якщо придивитися, то можна побачити що це не звичайний шматок картону, а на ньому зроблені прорізи таким чином, щоб з цієї картки можна було зібрати картонний інтерактивний стілець (мал.2.6.).



Мал.2.6. Інтерактивна взитна картка майстра з виготовлення меблів.

Дивлячись на усі вище перераховані проблеми та незручності використання візитних карток у тому вигляді що ми маємо, доцільно буде замислитись над тим що можна зробити за для того щоб щось змінити у кращу сторону, наприклад перевести їх у електронний вигляд. Для цього і було розроблено та реалізовано Web-додаток "Сервіс електронних візиток" з клієнтом для ОС Android, про який більш детально буде описано у розділі 4.

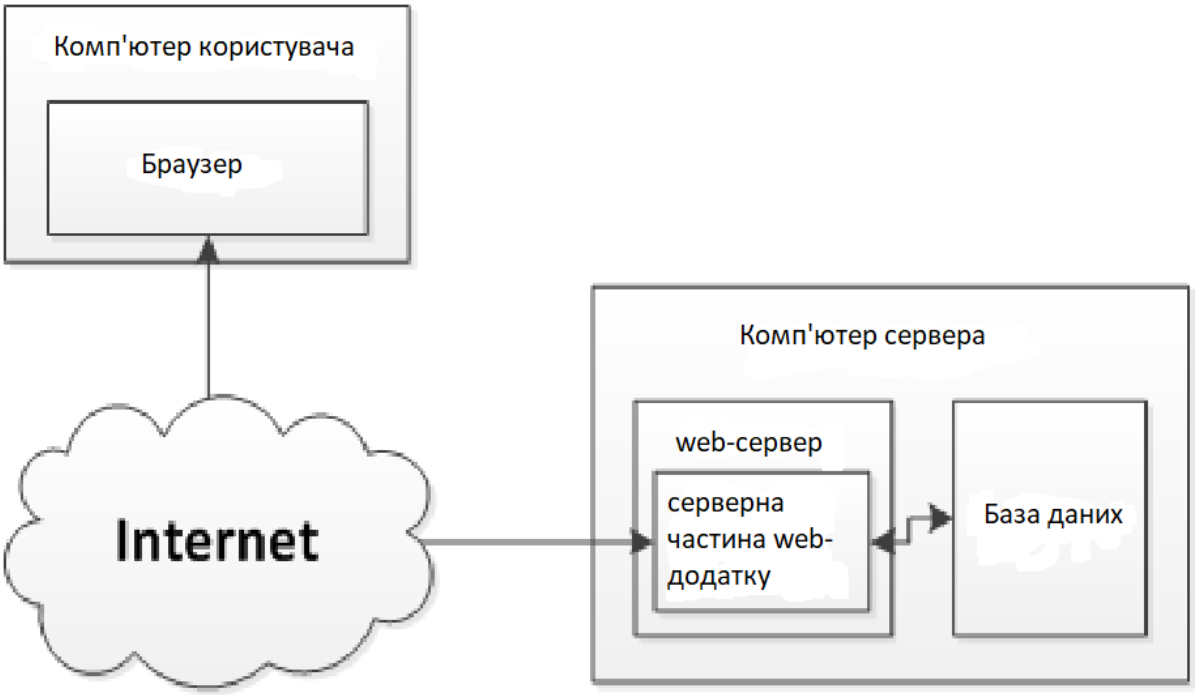
ВИСНОВОК ДО РОЗДІЛУ 2

У другому розділі було розглянуто специфіку користування та заберігання візитних карток у наш час, проблеми які виникають при використанні візитівок у її теперішньому вигляді та методи їх розв’язання.

Отже, можна зробити висновок, що візитні картки у їх сучасному вигляді не ефективні та застарілі, як метод поширення персональних контактних даних.

РОЗДІЛ 3. Розробка web-додатку на мові програмування JAVA. Основні етапи розробки та проектування

Як вже раніше згадувалось, web-додаток – це клієнт-серверний додаток, де клієнтом є будь-який браузер, а сервером – web-сервер. Задачі розподілені між клієнтом та сервером, зберігання інформації відбувається на сервері, зазвичай для цього використовуються бази даних. Взаємодія між сервером та клієнтом організовується зазвичай через мережу Internet. Схематичне зображення принципу роботи web-додатку можна побачити на мал. 3.1.



Мал.3.1. Схематичне зображення принципу роботи web-додатку.

3.1. Структура web-додатку

В цілому для забезпечення додатку, того щоб він мав право називатись повноцінним web-додатком, необхідно дуже не багато, а саме – це те щоб його структура складалася з 3-ох основнихчастин: фронтенду (клієнтської частини), бекенду (серверної частини) та місця де б зберігалася інформація (зазвичай –це база даних).

Клієнтська частина web-додатку - це графічний інтерфейс або те, що ви бачите на сторінці браузеру. Графічний інтерфейс відображається в браузері. Користувач взаємодіє з веб-додатком саме через браузер, виконуючи якісь дії (натискання на кнопки, перехід за посиланням).

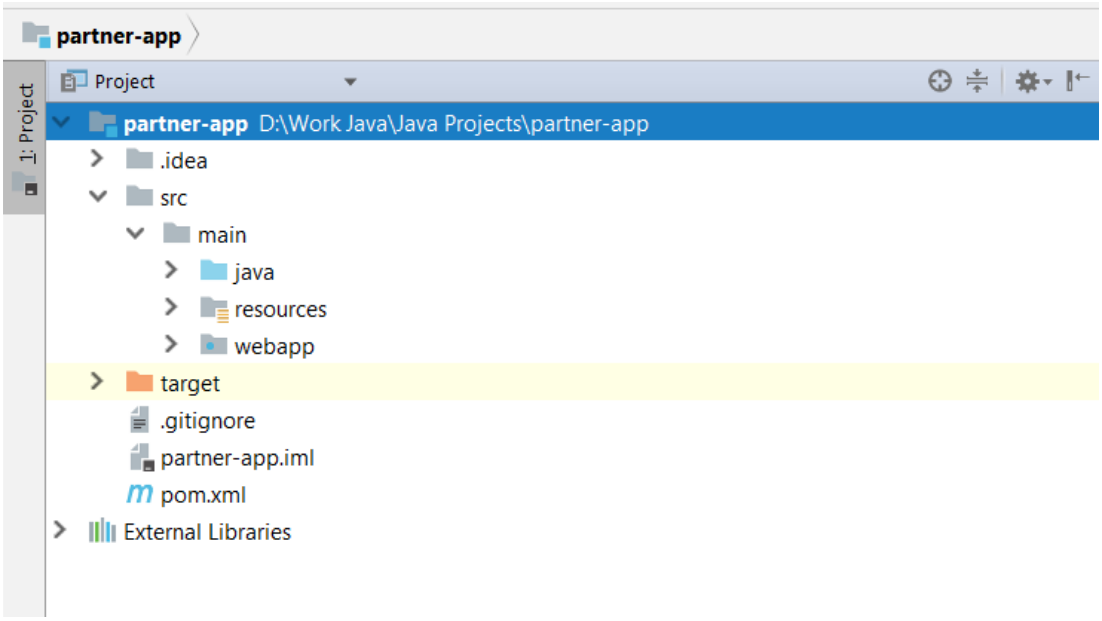
Серверна частина web-додатку - це програма або скрипт на сервері, що оброблює запити користувача (запити браузера). Найчастіше серверна частина web-додатку програмується на Java. При кожному переході користувача по посиланню браузер відправляє запит до сервера. Сервер обробляє цей запит, викликаючи деякий контроллер, який формує web-сторінку, описану мовою HTML, і повертає клієнту по мережі. Браузер відображає отриманий результат у вигляді нової сторінки.

База даних(система керування базами даних СКБД) – програмне забезпечення на сервері, що займається зберіганням даних і їх видачею в момент запиту. База даних знаходится (встановлюється) на сервері.

Спілкування клієнтута серверу web-додатку відбувається через HTTP-методи.

3.2. Проектування web-додатку

Перед тим як приступити до розробки було вирішено що при розробці web-додатку буде використовуватися зборщик проектів Maven, адже неможливо вручну контролювати усі процеси при розробці великих web-додатків. У середі розробки InteliJ IDEA було створено пустий maven-проект, який створив пустий проект з певною структурою і необхідними конфігураційними файлами (мал.3.2.).



Мал.3.2. Структура проекту.

Як можна бачити з мал.3.2. основними елементами структури додатку є 3 дерикторіх, а саме: java, з класами; resources,де зберігаються ресурси проекту накшталт конфігурації підключення до БД; webapp, де зберыгаються файли фронтенд частини, такы як сторынки web-додатку та стилы зы скриптами. Та фаіл pom.xml, це основний файл, який описує проект. Взагалі можуть бути додаткові файли, але вони відіграють другорядну роль.

Pom.xml складається з кореневого елементу та заголовку, залежностей та тегу <build>. Давайте детально розберемо для чого вони

необхідні.

Кореневий елемент та заголовок.

У Maven кожен проект ідентифікується парою groupId та artifactId. Щоб уникнути конфлікту імен, groupId - найменування організації або підрозділу і зазвичай діють такі ж правила як і при іменуванні пакетів в Java - записують доменне ім'я організації або сайту проекту. artifactId - назва проекту. Усередині тега version, як можна здогадатися зберігається версія проекту. Трійкою groupId, artifactId, version (далі - GAV) можна однозначно ідентифікувати jar файл програми або бібліотеки (мал.3.3.).

Також додається інформація, яка не використовується самим мавеном, але потрібна для програміста, щоб зрозуміти, про що цей проект:

<Name> … </ name> - назва проекту для людини

<Description> … </ Description> - Опис проекту

<Url> …</ url> - сайт проекту.

```
<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>partizan.global</groupId>
    <artifactId>partner-app</artifactId>
    <version>1.0</version>
    <packaging>war</packaging>

    <name>partner-app Maven Webapp</name>

```

Мал.3.3. Заголовок pom.xml.

Залежності

Залежності - наступна дуже важлива частина pom.xml - тут зберігається список всіх бібліотек (залежностей) які використовуються в проекті. Кожна бібліотека ідентифікується також як і сам проект - трійкою groupId, artifactId, version (GAV). Оголошення залежностей укладено в тег <dependencies> ... </ dependencies> (мал.3.4.).

```xml
<dependencies>

    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-core</artifactId>
        <version>5.1.8.RELEASE</version>
    </dependency>

    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-web</artifactId>
        <version>5.1.8.RELEASE</version>
    </dependency>

    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-webmvc</artifactId>
        <version>5.1.8.RELEASE</version>
    </dependency>
```

Мал.3.4. Приклад використання залежностей.

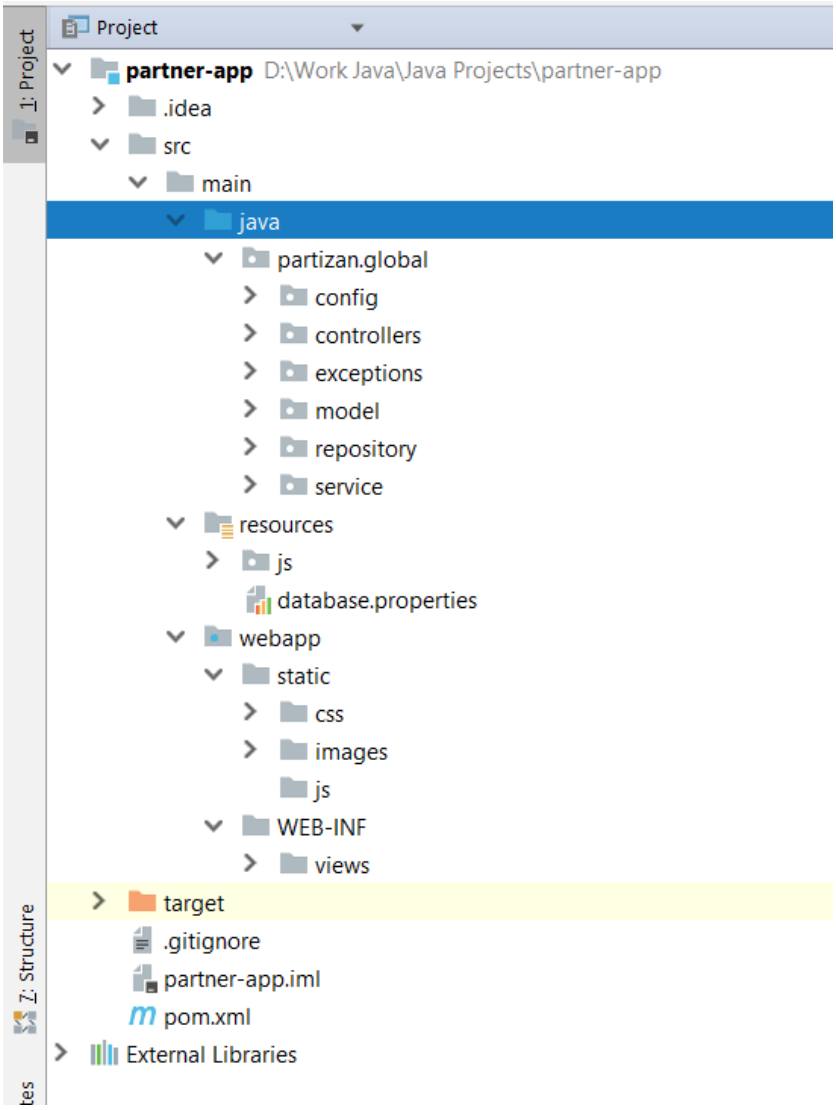
Тег <build>

Тег <build> не обов'язковий, так як існують значення за замовчуванням. Цей розділ містить інформацію о самій збірці: де знаходяться вихідні файли, де ресурси, які плагіни використовуються. Наприклад (мал.3.5.):

```xml
127     <build>
128         <finalName>ROOT</finalName>
129         <plugins>
130             <plugin>
131                 <artifactId>maven-war-plugin</artifactId>
132                 <version>2.4</version>
133                 <configuration>
134                     <failOnMissingWebXml>false</failOnMissingWebXml>
135                 </configuration>
136             </plugin>
137             <plugin>
138                 <groupId>org.apache.maven.plugins</groupId>
139                 <artifactId>maven-compiler-plugin</artifactId>
140                 <configuration>
141                     <source>1.8</source>
142                     <target>1.8</target>
143                 </configuration>
144             </plugin>
145         </plugins>
146     </build>
```

Мал.3.5. Приклад використання тегу <build>.

На малюнку 3.6. можна побачити повну структуру проекту, web-додатку «Сервіс електронних візиток».



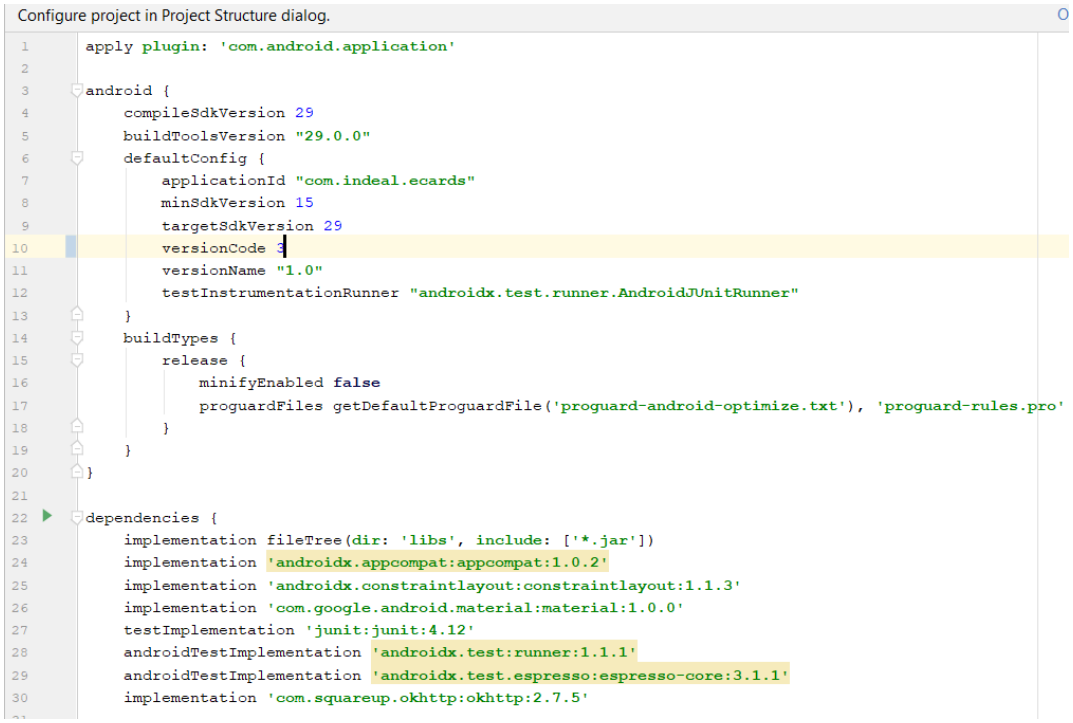
Мал.3.6. Повна структура проекту.

3.3. Проектування Android-додатку

При розробі додатку на Android, навідміну від web-додатку було обрано зборщик проектів Gradle, адже він більш підходить і користується популярністю бри зборці і розробці Android-додатків.

Gradle

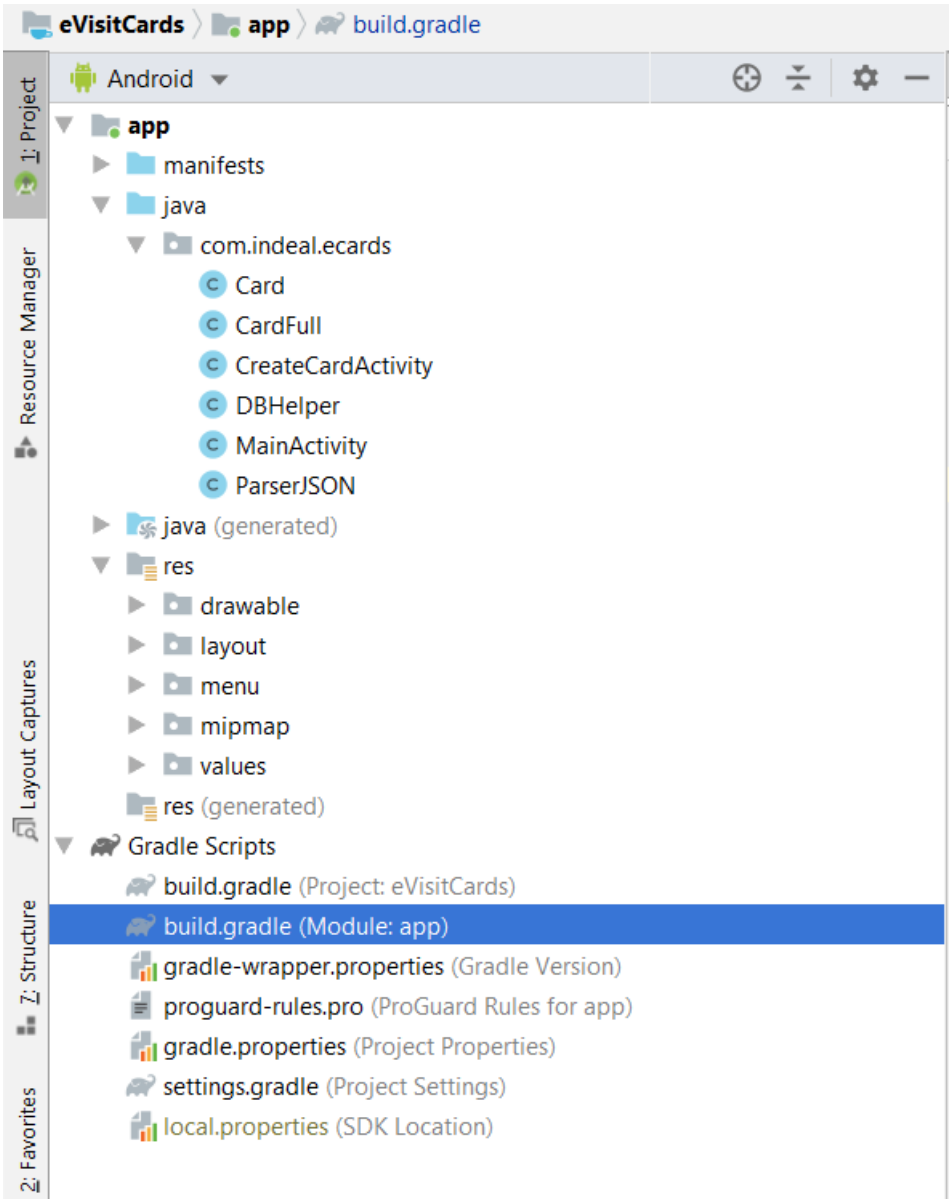
Gradle - система автоматичної зборки проектів, побудована на принципах Apache Ant і Apache Maven, але надає DSL на мовах Groovy і Kotlin замість традиційної XML-подібної форми подання конфігурації проекту (мал.3.7.).



Мал.3.7. Приклад структури конфігураційного файлу Gradle.

На відміну від Maven, заснованого на концепції життєвого циклу проекту, і Apache Ant, в якому порядок виконання завдань (targets) визначається залежностями (depends-on), Gradle використовує спрямований ациклічний граф для визначення порядку виконання завдань.

На малюнку 3.8. можна побачити повну структуру Android-додатку.



Мал.3.8. Структура Android-додатку.

3.4. Проектування бази даних web-додатку

Взагалі проектування бази даних, є одним із най важливіших кроків прои розробці будь-якого додатку що використовує БД. Для web-додатку було стоворено і приведен до третьої нормальної форми БД, що складається з 3-х таблиць: users, roles та partners. Розглянемо що ж таке нормалізація БД або ж приведення БД од однієї з нормальних форм.

Нормалізація бази даних — процес розбиття одного відношення (таблиці у БД) відповідно до вимог нормальної форми БД на декілька відношень на базі функціональних залежностей інакше кажучі розбивка однієї таблиці на декілька пов'язаних.

Нормальна форма — це сукупність вимог до таблиць БД, за для усунення з бази надлишкових функціональних залежностей між атрибутами інакше – це сукупність вимог, яким має задовольняти відношення.

Перша нормальна форма

1НФ (1NF) потребує задовільненню умов:

- Кожна таблиця повинна мати первинний ключ: мінімальний набір колонок, які ідентифікують запис.
- Уникнення повторень груп правильно визначаючи неключові атрибути.
- Атомарність: кожен атрибут повинен мати лише одне значення.

Друга нормальна форма

2НФ (2NF) потребує, щоб дані, що зберігаються в таблицях із композитним ключем (або ще складеним чи вторинним), не залежали лише від частини ключа:

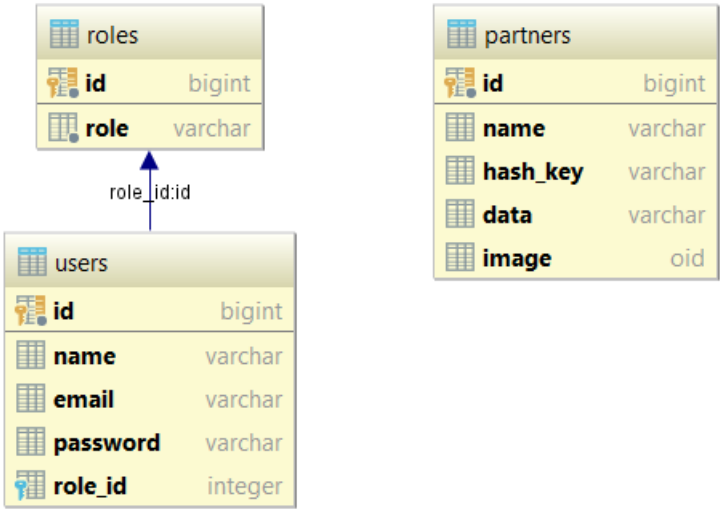
- Схема бази даних повинна відповідати вимогам першої нормальної форми.

- Дані, що повторно з'являються в декількох рядках, виносяться в окремі таблиці.

Третя нормальна форма

- ЗНФ (3NF) потребує, щоб дані в таблиці залежали винятково від первинного ключа:
- Схема бази даних повинна відповідати всім вимогам другої нормальної форми.
- Будь-яке поле, що залежить від первинного ключа та від будь-якого іншого поля, має виноситись в окрему таблицю.

На малюнку 3.9. показано таблиці бази даних web-додатку "Сервіс електронних візиток", зі зв’язками між ними.



Мал.3.9. Таблиці бази даних.

ВИСНОВОК ДО РОЗДІЛУ 3

У цьому розділі було розглянуто та розібрано специфіку створення web-додатку на мові Java, створення Android-додатку а, також етапи проектування БД.

Так як у web-додатків є і свої мінуси, головний з яких – це те що для роботи веб-додатку потрібне постійно з’єднання з всесвітньою мережею інтернет, і якщо головним функціоналом web-додатку є щось, що не потребує постійного доступу до інтернету або щось, що є дуже ресурсоємким, то в цьому випадку більш правильним рішенням буде розробити десктопний додаток. Через це і виникла потреба розробки додаткового Android-додатку, який буде являтись клієнтом існуючому web-додатку.

Тому не можна сказати що web-додаток є кращім за десктопний додаток або навпаки, адже є такі моменти де перший краще за другого, а є де навпаки.

РОЗДІЛ 4. Web-додаток «Сервіс електронних візиток» з клієнтом для ОС Android

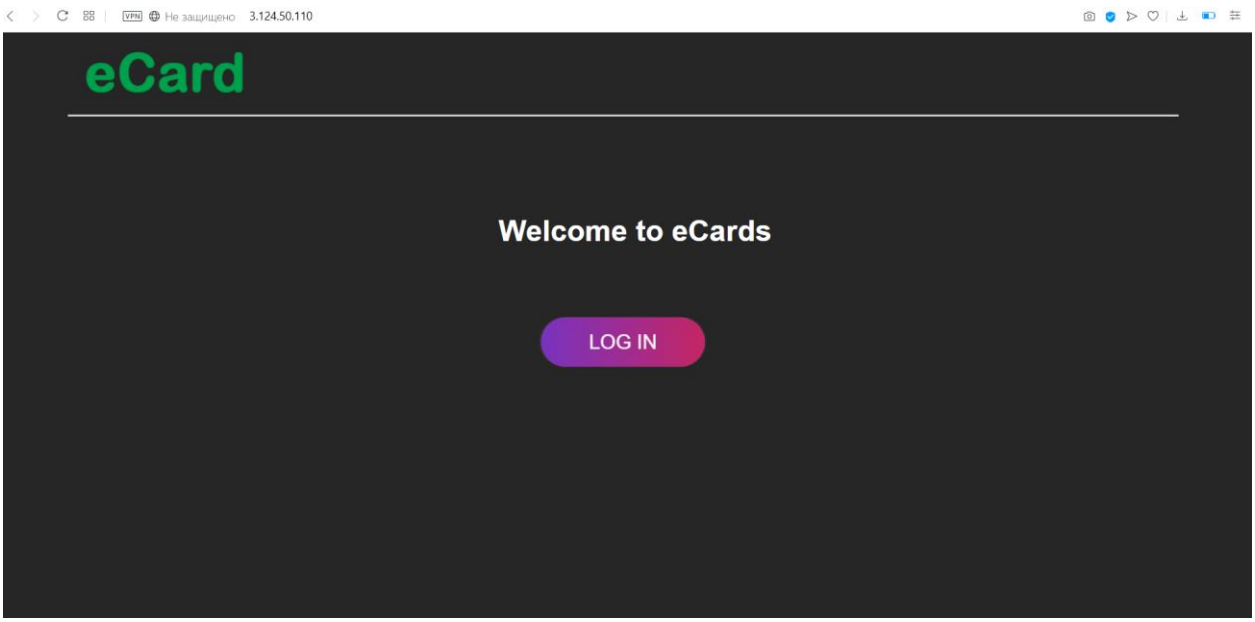
4.1. Прзначення web-додатку «Сервіс електронних візиток»

Відштовхуючись від недоліків і складностей що виникають при використанні паперових, чи взагалі будь-яких візитних карток у фізичному вигляді, доцільно було розглянути альтернативні варіанти, які дозволять зручно, менш затратно та з меншою шкодою для природи, організувати використання візитних карток.

Основною ціллю розробки web-додатку «Сервіс електронних візиток» з клієнтом для ОС Android, було змінити сучасне уявлення про візитні картки, розробивши додаток, який буде забезпечувати змогу користувачеві цим додатком створювати, редагувати та ділитися з іншими коричтувачами контактними даними, лише налавши посилання на свою чи відому йому візитну картку, або ж просто сказати унікальний код, що має кожна візитка, створена у даному додатку.

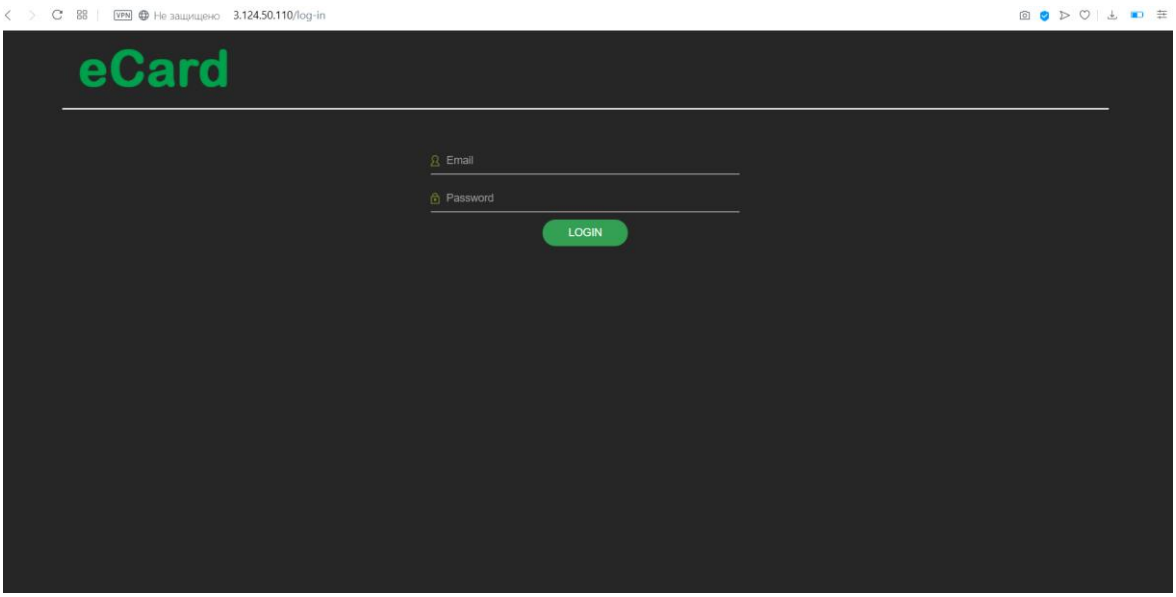
4.2. Принцип роботи web-додатку «Сервіс електронних візиток»

Головна сторінка web-додатку уявляє собою сторінку що пропонуєлюдині авторизуватися у даному додатку, для подальшої роботи з ним (мал.4.1.).



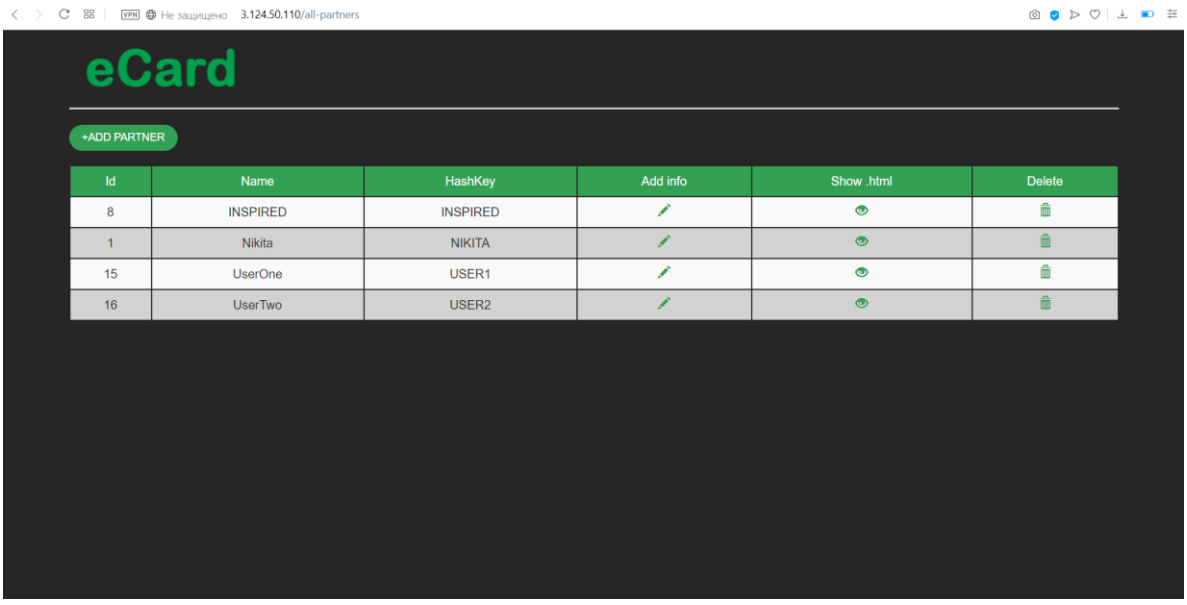
Мал.4.1. Головна сторінка web-додатку «Сервіс електронних візиток eCard»

З головної сторінки кормстувач може, натиснувши кнопку «LOG IN», перейти до форми авторизації, де ввівши корректні параметри авторизації потрапити до самого додатку та мати змогу почати користуватися ним (мал.4.2.).



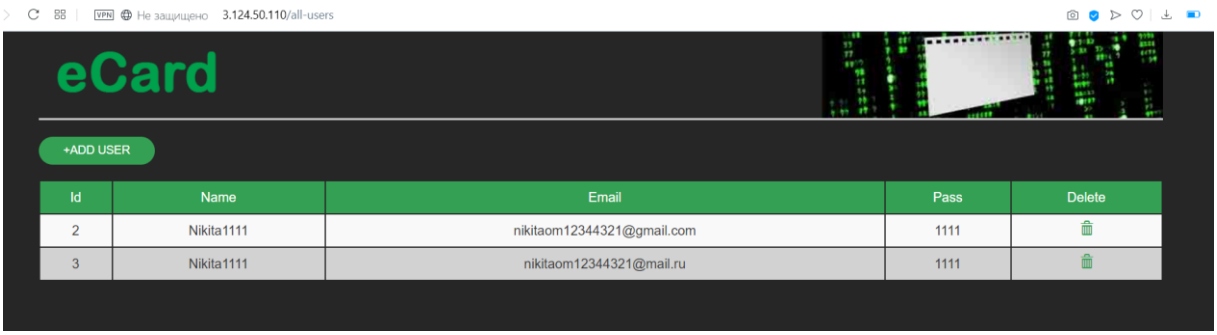
Мал.4.2. Сторінка авторизації.

Авторизувавшись у додатку, користувач отримує можливість подальше користуватись додатком, а саме створювати, редагувати, переглядати та видаляти візитівки. Авторизувавшись, користувач отримує сторінку, де він може бачити усі існуючі записи (мал.4.3.), здекилькома кнопками керування.



Мал.4.3. Сторінка зі списком існуючих записів.

На теперішній момент, дана сторінка відкрида для адміністрування усіс користувачам додатку, але це є не зовсім корректним кроком адже зараз будь-який користувач шо потрапив на дану сторінку має права на редагування та видалення не тільки своїх, а і сторонніх записів, таму для додатку вже розроблюється адміністративна панель (мал.4.4.), де редагувати та видаляти записи усіх користувачів зможуть лише адміністратори даного додатку.

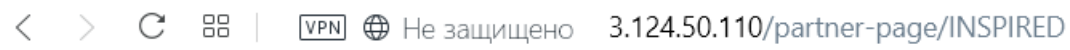


Мал.4.4. Одна зі стотрінок адміністративної панелі керування.

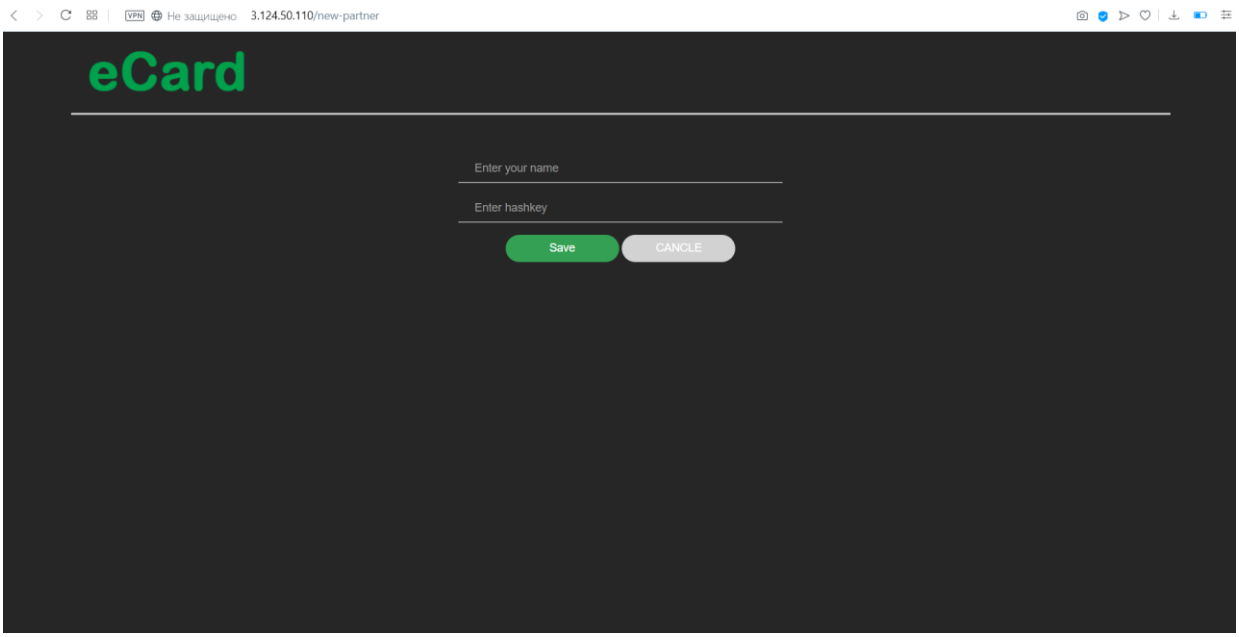
Отже, як вже казалося вище, зі сторінки (иал.4.3.) користувачеві надається змога зокрема:

- Створити новий запис, який фактично в подальшому і буде формувати зміст електронної візитної картки. Це можна зробити натиснувши кнопку натисканн «ADD PARTNER», після чого користувача буде перенаправлено на сторінку з початковою формою створення запису (мал.4.6.) (електронної візитної картки), з формою що складається з 2-х полів: наіменування запису (може бути будь-яким, воно ніде не відображуватимется у подальшому, лише у авдмін панелі), та унікального хеш-ключа, який у подальшому буде формувати посилання на створювану

візитну картку в інтернеті (остання запис після знаку “/” у стоці пошуку / url-адреса)(мал.4.5.) та буде яввлятися унікальним ключем для отримання візитної картки у додатку на Android пристрої.



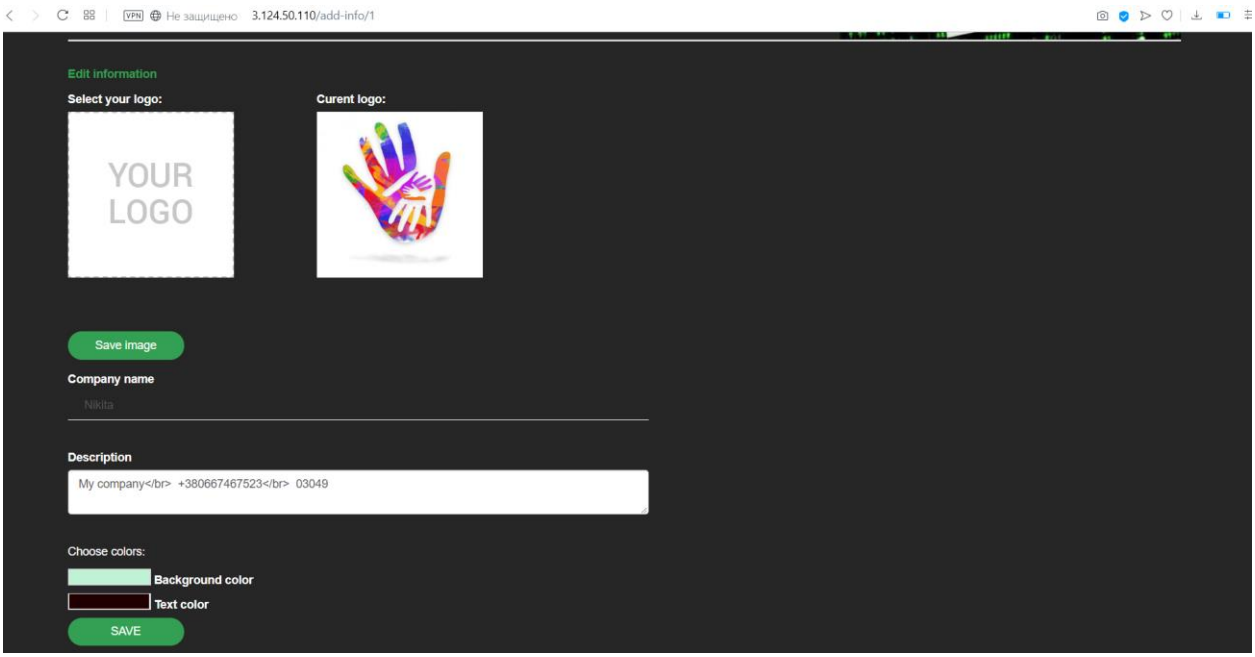
Мал.4.5. Приклад формування прсилання на певну візитну картку в інтеренті з унікальним хеш-ключем «INSPIRED».



Мал.4.6. Сторінка з формою для створення нового запису.

- Відредагувати запис. Натиснувши на кнопку з логотипом олівця, у стовбчику Add info (мал.4.3.) куристувачеві надасця змога відредагувати запис, а саме створити електронну візитну картку, додавши логотип (чи змінивши цого, за наявності), написати ім’я

або назву організації на кардці, додати додаткову інформацію, притаманну візитним карткам, таку як, контактні данні (адресу, номер телеону,арресу електронної скриньки тощо.), за бажанням опис того, які послуги предоставляє людина чи компанія, якою була створена, та поширена данна візитна картка (мал.4.7.), у зручном та доволі гнучкому форматі редагування, де дозволяється використання тегів з мови розмітки HTML. А також трохи стилізувати візитку, обрафши її колір та колір тексту на ній (мал.4.7.).

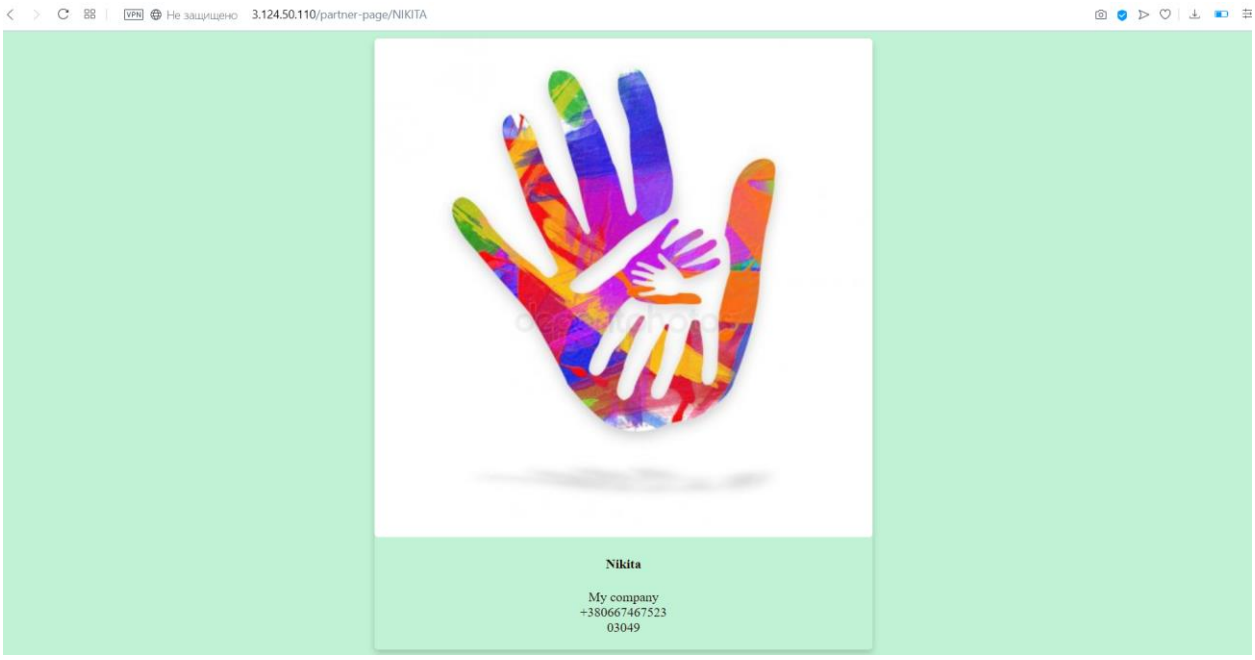


Мал.4.7. Сторінка з формою редагування візитної картки.

На даній сторінці користувач може бачити одаразу ж існуючу інформацію (наприклад є змога бачити попередньо-використовуваний логотип у правій частині), та вже відштовхуючись від цих даних приймати рішення, яка інформація потребує редагування, а яка повинна лишитися як є.

При натисканні на кнопу редагувати форма редагування віддається користувачеві з усіма данними, що були раніше внесені до неї, за для більш комфортної та зручної роботи, без необхідності згадувати, якими данним була вона заповнена раніше.

- Подивитися на фінальний вигляд візитної картки, та отримати url-адресу поточної елетронної візитки, для змоги у подальшому поширювати її. Електронна фізитка має мінімалістичний, стардантизований вигляд, де відмінним будуть лише наповнення (інформація, кольори фону та тексту, та зображення), а в загальному вигляд усіх карток буде однаковим (мал.4.8.).



Мал.4.8. Приклад вигляду електронної візитної картики у web-додатку.

- Видалення запису та усієї єлектронної візитки.

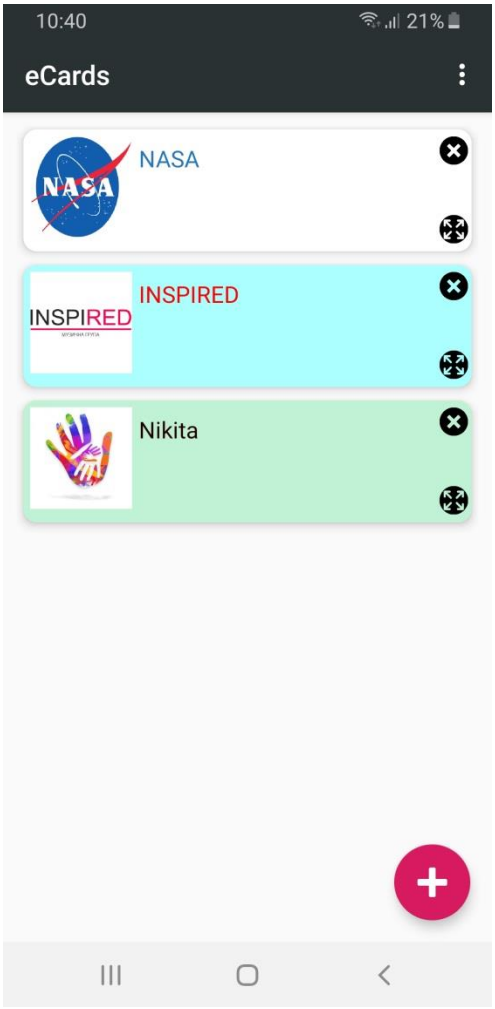
Також web-додаток надає API, для отримання вызитної картки у форматі JSON, для її використання мобільним додатком на Android. А саме у JSON форматі передається: зображення (логотип на візитній кардці), у перекодованому у формат base64 вигляді, перекодування відбувається на бекенді; назва компанії або ж ім’я власника картки, опис, контактні данні та елементи персоналізації у дизайні, а саме кольори фону та тексту. Приклад даного запиту з поверненою у ворматі JSON строкою можна побачити нижче (мал.4.9.).

Мал.4.9. Строка у форматі JSON, що на дає API web-додатку.

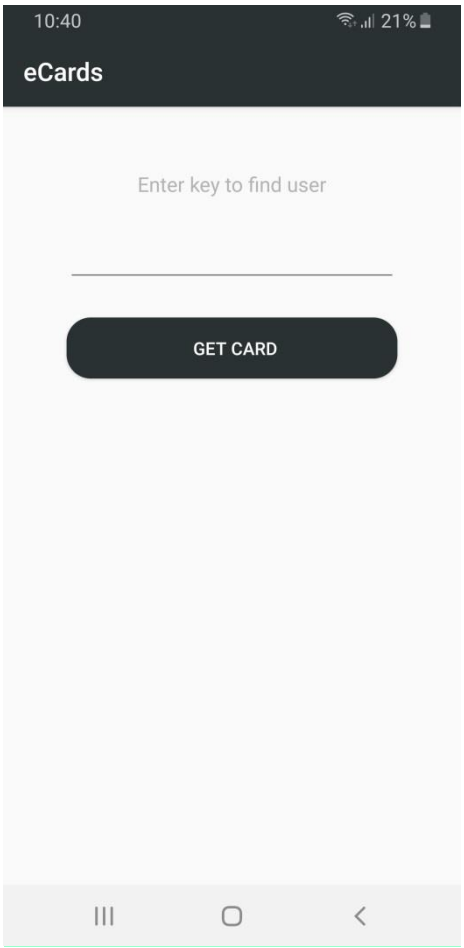
4.3. Пинцип роботи Android-додатку, його призначення

Оже розглянемо сам додаток під операційну систему Android, який є клієнтом web-додатку eCard, для якого саме і було розроблено вище вказаний API.

Головна сторінка додатку представляє собою список візитних карток, які користувач додав собі, або ж пусту сторінку з рожевою кнопкою “+” (мал.4.10.), при натисканні на яку користувача буде перенаправлено на іншу сторінку, на якій він зможе ввівши унікальний хеш-ключ, який поширює власник електронної візитки, отримати собі дану візитку (мал.4.11.).

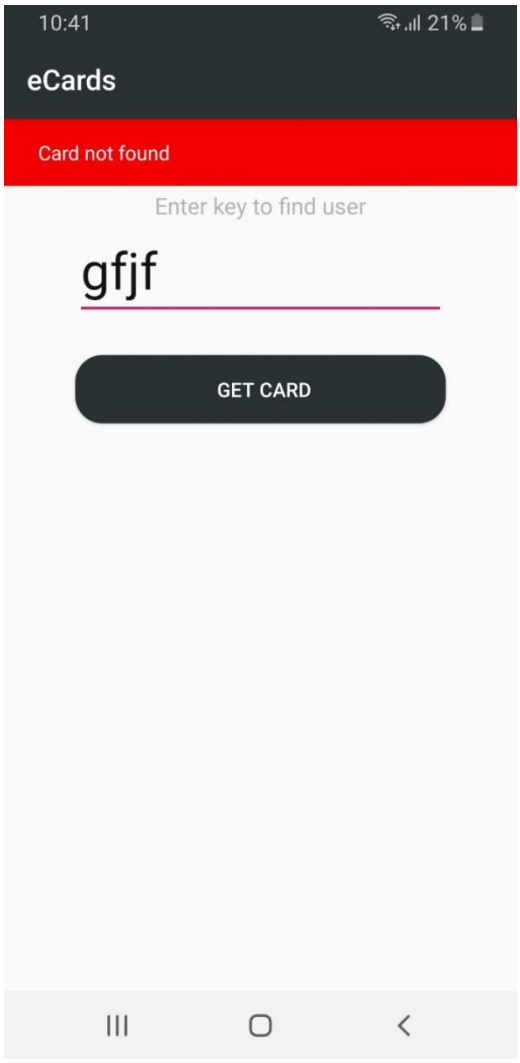


Мал.4.10. Головне вікно Android-додатку, з вже доданим до списку трьома візитками.



Мал.4.11. Вікно зполем для вводу унікального ключа, для отримання за ним візитної картки.

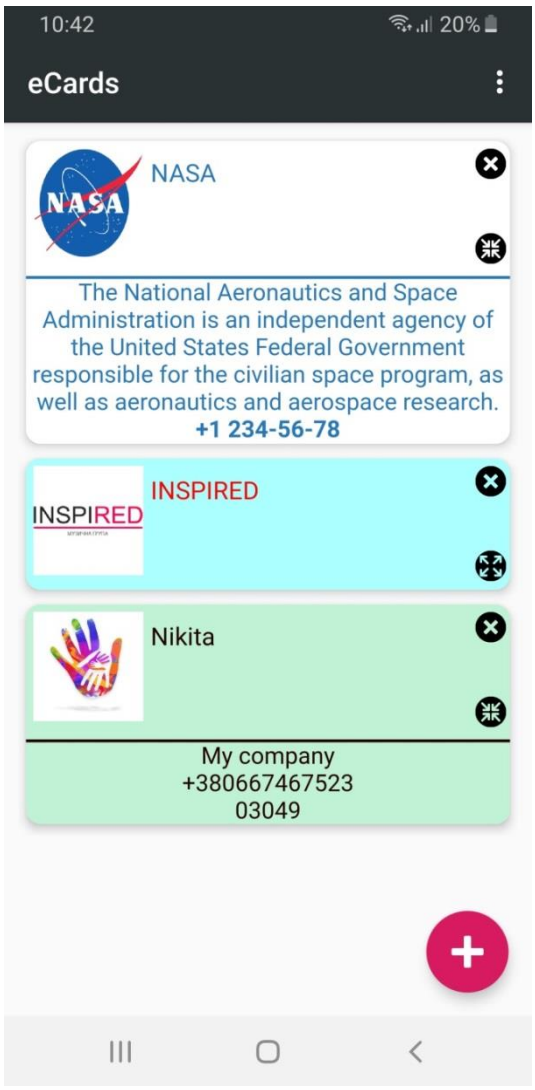
Функціонал додатку розроблено таким чином, що наразі введення некоректного (неіснуючого) ключа, аобо ж за відсутності інтернетз'єднання при спробі додати нову візитну картку до свого списку, користувачеві буде видано швидку повідомлення з відповідним текстом, створено інструментом Snackbar, про неможливість створити картку та запропоновано повторитиопераційю з корректними даними чи при наявності зв'язку з мережею Internet (мал.4.12.).



Мал..4.12. Приклад повідомлення при введенні некоректних даних.

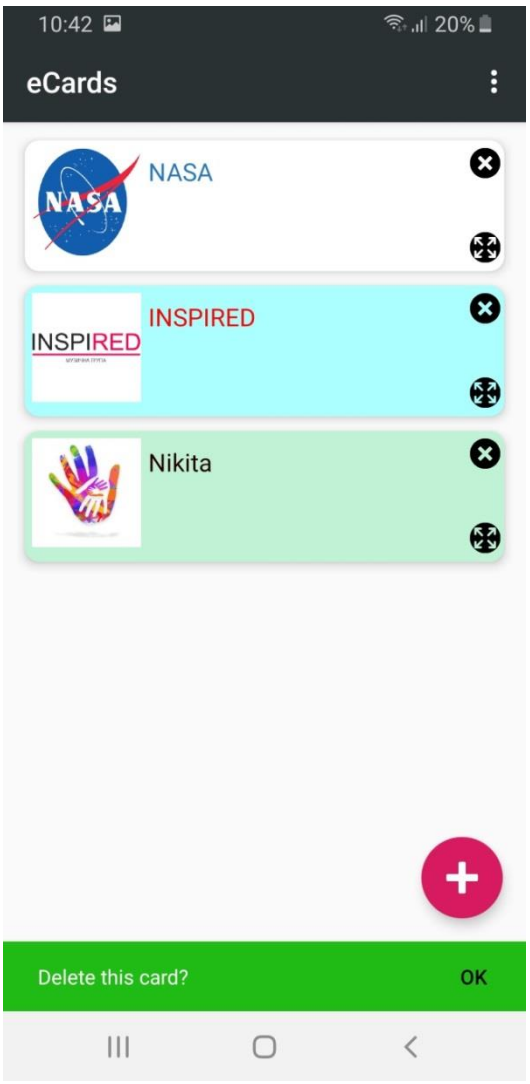
Наразі, якщо введені данні будуть коректні та буде зв'язок із мережею інтернет, то з додатку буде зробрено запит на посилання що надає API web-додатку, та отримано дані, з яких буде створено електронно візитну картку, яка зберігатимется на даному Android-пристрої, у локальній БД (SQLite), а користуавча перекине доголовного екрану додатку, де він одразу ж побачить щойно отриману та створену електронну візитну картку.

Початковий вигляд візитної картки, не несе у собі усієї інформації, закладеної в неї її створювачем, а містить лише логотип та ім'я/назву компанії. Для бвльш детального перегляду усієї інфориації, користувач може натиснути на кнопку, яка розміщена на усіх картках, з правої сторони, у нижній частині, тоді картка розвернется у тому ж вікні, або просто натиснувши на картку, тоді відбудется те ж саме, аналогічним чином, натиснувши на дану кнопку, коли картка буде вже розвернута (при розвертанні картки, кнопка змінюється на іншу, що нагадує жест згортання), картка знову прийме свій початковий, компактний вигляд (мал.4.13.). На малюнку нижче можна наглядно порівняти картки у розгорнутому виглядв (це 1 та 3), та згорнуту (2), а також зображення на їх кнопках.



Мал.4.13.Приклад картки у розгорнутому вигляді.

Також на кожній картці присутня кнопка з хрестиком, з права,у верхньому куті, при натисканні на яку, коримтувач матиме змогу видалити запис електронної візитної картки зі свого Android-пристрою (картка буде існуавти у БД web-додатку, видалення лише з внутрішньої бази даних Android-додатку), попередньо підтвердивши свою дію у спливаючій стрічці з низу екрану, для запобігання випадкового видалення (мал.4.14.).



Мал.4.14. Приклад підтвердження видалення картки з пристрою.

4.3. Доступ до сервісу

Web-додаток "Сервіс електронних візиток" з клієнтом для ОС Android можна знайти у вільноиу доступі у мережі Internet, а саме: web-додаток, було розміщено на серіері Amazon там само знаходится і база даних, де зберігаються усі існуючі візитні картки.

Отримати доступ до web-додатку на момент написання роботи можна перейшовши за посиланням: *http://3.124.50.110* , де додавши до даної url-адрест різні ендпоінти (кінцеві точки посилання), можна потрапити на різні сторінки додатку, наприклад:

- http://3.124.50.110/log-in - для авторизації;
- http://3.124.50.110/all-users - для перегляду усіх користувачів;
- http://3.124.50.110/all-partners - для перегляду усіх записів;
- http://3.124.50.110/new-partner - для створення нового запису;
- http://3.124.50.110/add-info/{унікальний ключ користувача картки, яку необхідно відредагувати} –для редагування запису;
- http://3.124.50.110/partner-page/{унікальний ключ користувача картки, яку переглядаєте} –для отримання візитнох картки.

Отримати доступ до додатку на Android можна відвідавши магазин застосунків від Google – Google Play, за назвою: «eCard»; або ж за посиланням:
https://play.google.com/store/apps/details?id=com.indeal.ecards .

ВИСНОВОК ДО РОЗДІЛУ 4

Отже, в даному розділі було наглядно розібрано принцип користування web-додатком та додатком на Android, а також для чого вони призначені, які задачі вирішують та порівняно практичність, користування фізичними та електронними візитними картками.

При порівнянні використання звичайних візиних карток та користування сервісом електронних візиток, можна наглядно побачити та зрозуміти перевагу сервісу, завдяки його більш ефективній роботі, яка полягає у тому, що немає неоюхідності носити з собою, згадувати де знаходятся, та вринципі не нищити природу за для розповсюдження своїх контактних даних, достатньо лише отримати посилання, чи додати картку в електронному вигляді до списку карток у своєму Android-додатку.

РОЗДІЛ 5. Сервіси по розміщенню web-додатків та додатків під ОС Android, для доступу до них зі всесвітньої мережі Internet

5.1. Розміщення web-додатку та БД

Для забезпечення постійного доступу до web-додатку у мережі інтернет, та для можливості завантажити додаток на Android-девайс, вони були розміщені у всесвітній менежі Internet.

Web-додаток було розмішено на віддаленому сервері від Amazon (AWS) (мал.5.1.).

Мал.5.1. Офіційний логотип сервісу AWS від Amazon.

Amazon Web Services (AWS) - це найпоширеніша в світі хмарна платформа з широкими можливостями, що надає більше 175 повнофункціональних сервісів для центрів обробки даних по всій планеті. Мільйони клієнтів, в тому числі стартапи, які стали лідерами за

швидкістю зростання, найбільші корпорації і передові урядові установи, використовують AWS для зниження витрат, підвищення гнучкості і прискореного впровадження інновацій.

Для початку користуавння консоллю AWS (мал.5.2.), необхідно було створити то оплатити аккаунт. Наступним кроком буде рати необхідну кількість ресурсів, які потенційно буде використовувати додаток. Дале встрановити операційну систему та налаштувати оточення, а саме у даному випадку, в якості операційной системи було обрано CentOS.

Проект CentOS – це спільна програма вільного програмного забезпечення, орієнтована на спільноти та на забезпечення надійної екосистеми з відкритим кодом навколо платформи Linux.

CentOS Linux - це послідовна, керована платформа, яка підходить для широкого спектру розгортань. Для деяких громад з відкритим кодом це міцна передбачувана база, на якій потрібно будувати.

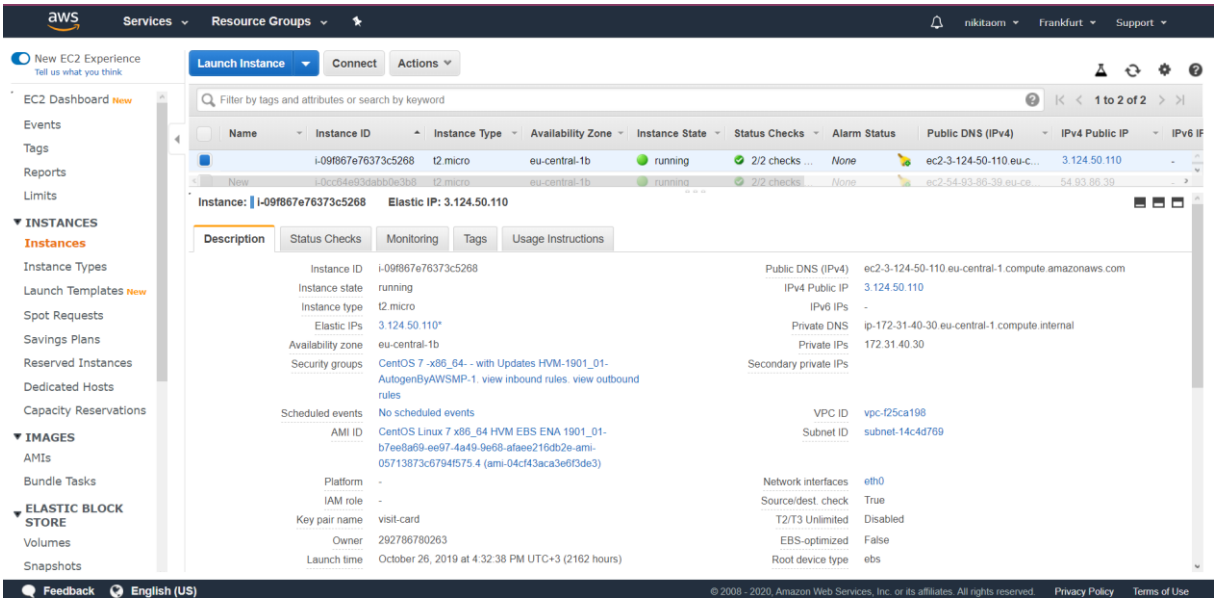
Далі на сервер (віддалений комп'ютер) будо встановлено Java 1.8, Tomcat 10 та налагоджено Nginx.

Nginx – це HTTP-сервер та зворотний проксі-сервер, поштовий проксі-сервер, а також TCP / UDP проксі-сервер загального призначення, спочатку написаний Ігорем Сисоєва (мал.5.3.). Вже тривалий час він обслуговує сервери багатьох високонавантажених сайтів, таких як Яндекс та Mail.Ru. Згідно зі статистикою Netcraft nginx обслуговував або проксирував 25.44% самих навантажених сайтів в січні 2020 року. Ось деякі приклади успішного впровадження nginx: Dropbox, Netflix, Wordpress.com, FastMail.FM.

Наступним кроком було виділення серверу динамічної IP адреси, адже за замовчанням була налагоджена лише статична.

Також завдяки дуже гнучким можливостям консолі AWS, було створено та «піднято» БД, яку для більш безпечного використання, було встановлено

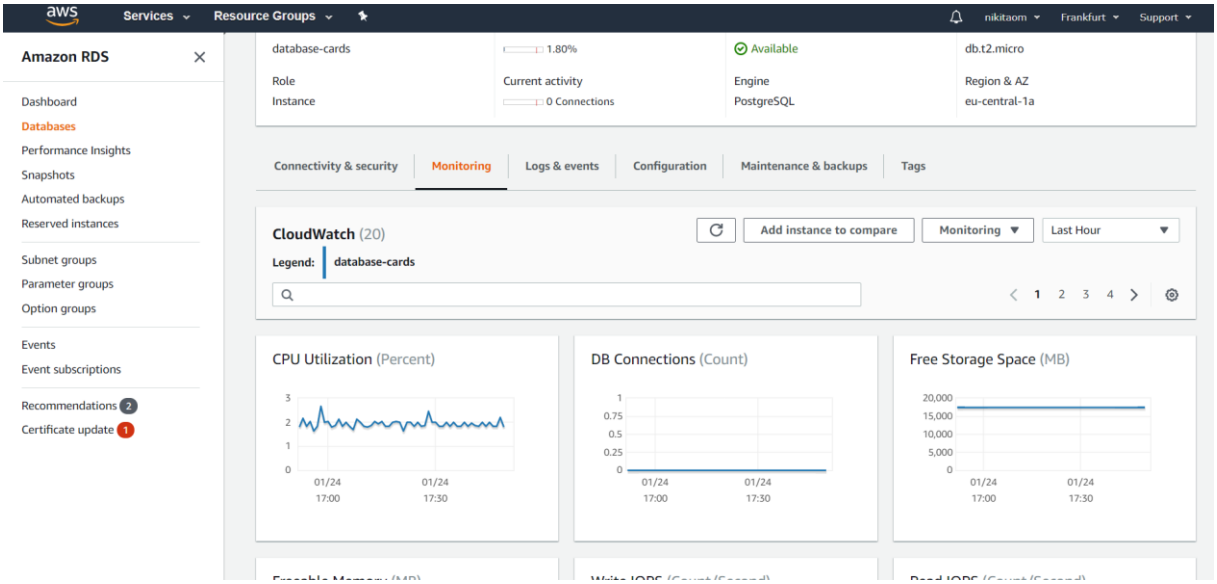
окремо від web-додатку за для забезпечення постійної та безперебійної роботи з БД, не звадаючи на чтомн серверу з додатком, тобто, навіть якщо додаток перестане працювати («впаде» сервер), доступ до БД будезалишатися, і буде змога нею користуватись. Дана практика є дуже популярною в наш час через свою безпеку та надійність.



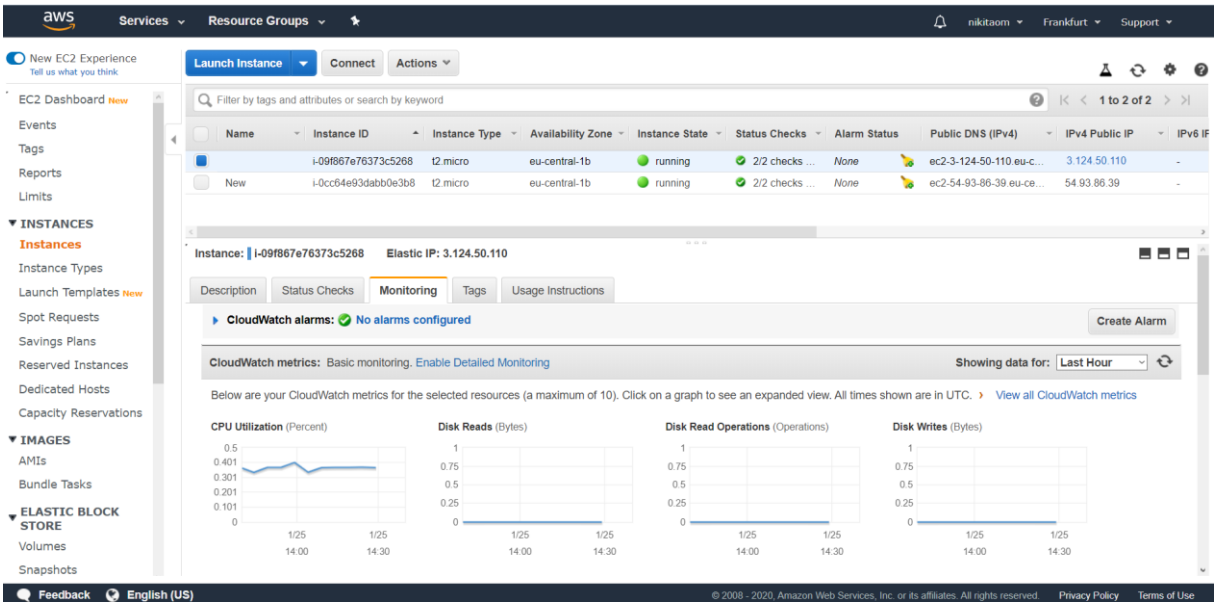
Мал.5.2. Відкрита AWS консоль web-додатку.

Мал.5.3. Логотип Nginx.

Також AWS консоль надає дуже зручні інструменти з контролю за ресурсами та керування данними (мал.5.4.). Завдяки чому можна оцінювати наскільки ефективним був вибір певних налаштувань (наприклад віиділеної кількості оперативної пам’яті), та відштовхуючись від цього приймати рішеннявносити певні коррективи або ж ні.



Мал.5.4. Сторінка для моніторингу за використанням ресурсів БД.



Мал.5.5. Сторінка моніторингу піднятих серверів.

5.2. Розміщення Android-додатку

Android-додаток було розміщено на одном з найпопулярніших офіційних електронних магазинів додатків від Google – Google Play (мал.5.6.).

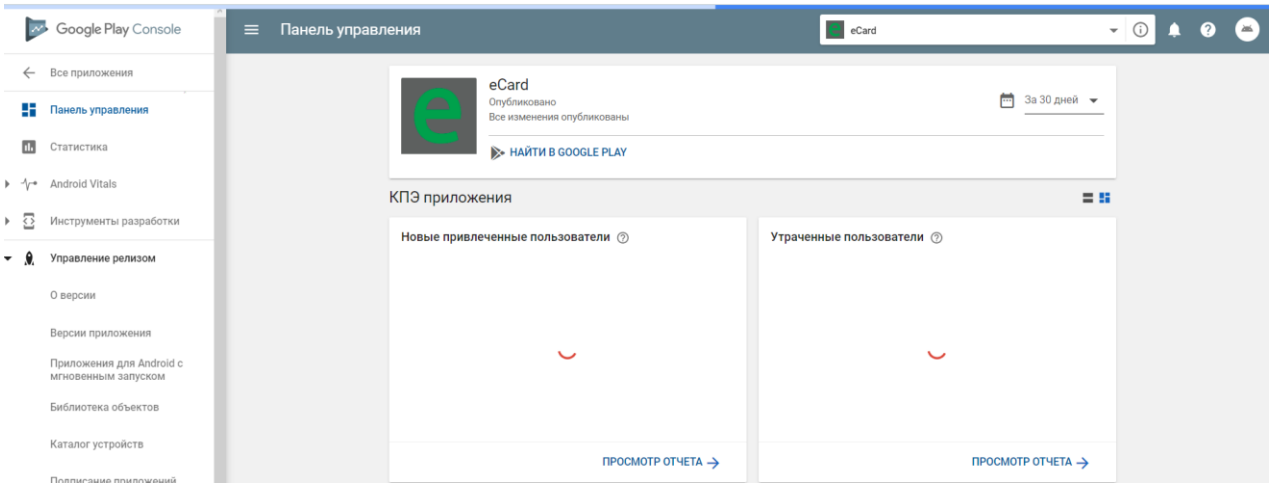
Мал.5.6. Логотип магазину додатків від Google – Google Play.

Для забезпечення можливості будь-кому, хто має пристрій під управлінням ОС Android, та хоче користуватися додатком на своєму пристрої, додаток було розмішено в магазині додатків від Google – Google Play.

Для цього було створено аккаунт у Google, та верифіковано його, шляхом заповнення певної персональної інформації, про користувача та подальшою його оплатою. Після чого стало можливим додавати свої застосунки до маркету. Google Play має зручну та зрозумілу консольрозробника, де можна створити проект, додавши назву, опис, логотипи та іконки додатку, пройти тест на те, якій категорії людей підходить додаток, обрати рубрику, додати скріншоти та розмістити файл зі

своєю програмою. Даний файл має бути зкомпільований певним чином, так як потребує Google Play Console. Далі після того як додаток буде збережено, він матиме пройти верефікацію від Google, і якщо задовільнятиме усім вимогам, то на протязі від 1 да 7 днів, він буде опублікований у Google Play. Після чого будь-хто матиме змогу завантажии його собі на пристрій.

Далі все що відбувається з додатком можна відстежувати у Google Play Console (мал.5.7.).



Мал.5.7. Відкрита Google Play Console.

5.3. Способи з’єднання з віддаленими серверами, та ПЗ для цього

Незважаючи на те що web-версія консолі AWS надає дуже широкий функціонал користувачеві, все одно для того щоб мати змогу розмістити додаток, запустити/зупинити чи перезавантажити віддалений сервер, о бо ж просто налаштувати на ньому оточення, необхідно отримати до нього доступ. Для цього в консолі AWS було отримано логіни та паролі (в вигляді файлів-ключів), для з’єднання з сервером через SSH.

SSH (Безпечна оболонка – Secure Shell) – це криптографічний мережевий протокол для безпечної роботи мережевих служб через незахищену мережу. Типові програми включають віддалений командний рядок, вхід та віддалене виконання команд, але будь-яка мережева служба може бути захищена SSH.

SSH забезпечує захищений канал у незахищеній мережі в архітектурі клієнт-сервер, з'єднуючи клієнтську програму SSH з SSH-сервером. Специфікація протоколу розрізняє дві основні версії, які називаються SSH-1 та SSH-2. Стандартний порт TCP для SSH становить 22. SSH зазвичай використовується для доступу до операційних систем, схожих на Unix, але він також може використовуватися в Microsoft Windows. Windows 10 використовує OpenSSH в якості свого клієнта SSH за замовчуванням.

SSH був розроблений як заміна для Telnet і для незахищених протоколів віддаленої оболонки, таких як протоколи Berkeley, rrsin, rsh і rexec. Ці протоколи надсилають інформацію, зокрема паролі, у простому тексті, роблячи їх сприйнятливими до перехоплення та розкриття за допомогою аналізу пакетів. Шифрування, яке використовується SSH, призначене для забезпечення конфіденційності та цілісності даних у незахищеній мережі, наприклад, в Інтернеті, хоча файли просочуються

Едвард Сноуден вказує, що Агенція національної безпеки іноді може розшифровувати SSH, дозволяючи їм читати вміст сесій SSH.

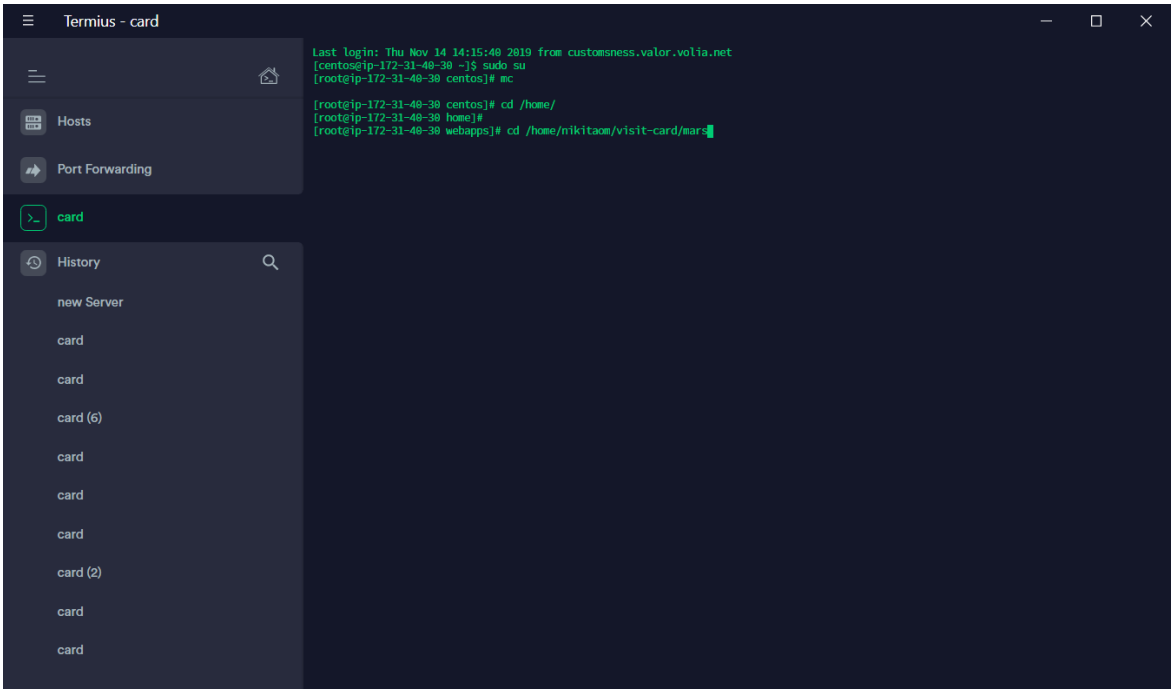
В якості SSH клієнта було обрано програму Termius (мал.5.8.).

Мал.5.8. Офіційний логотип Termius.

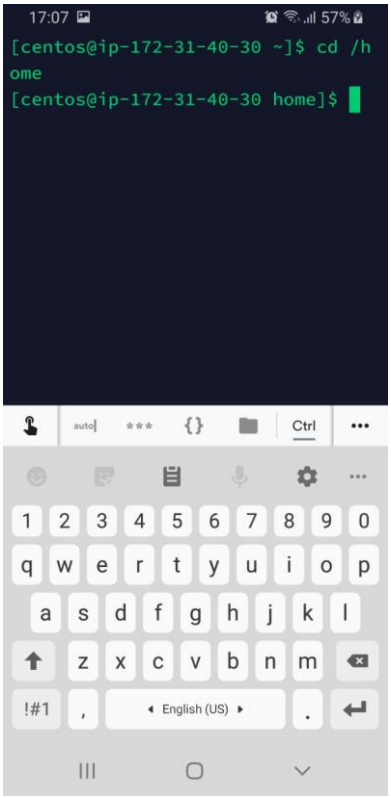
Як заявляють самі розробники, Termius – це не просто клієнт SSH, це повне рішення командного рядка. Йог обуло розроблено для безпечного отримання доступу до пристроїв Linux або IoT зі свого мобільного пристрою Android або iOS, а також будь-якого комп’ютера Windows, macOS або Linux. Він сумісний з Mosh, що забезпечує відмінну надійність при високо затримкових з'єднаннях, що постійно змінюються.

Дане ПЗ було обрано за його дуже розвиненй і велику спільноту, де можна знайти ішення багатьох типових проблем, а також за те що його можна всановити на будь-який пристірій, зокрема і на мобільний телефон, що дуже облегшує доступ та контрось за сервером з буль якого місця, де є інтернет, але немає персонального комп'ютеру. Приклад з'єднання по SSH, та вигляд інтерфейсу користувач можна побачити на мал.5.9. (доступ з

персонального комп’ютеру) та мал.5.10. доступ з Android-пристрою.

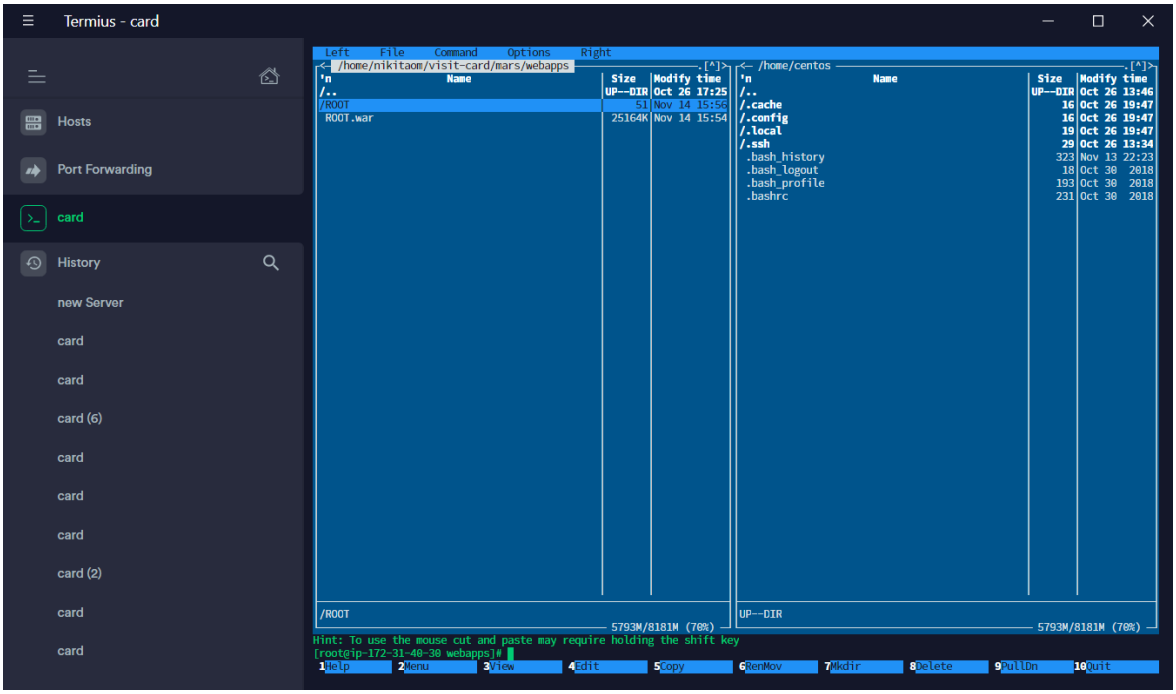


Мал.5.9.. Термінал Termius відкритий з ПК.



Мал.5.10.. Термінал Termius відкритий з пристрою на ОС Android.

Також дуже великим плюсом постійної підтримки та розвинення розробниками даного ПЗ є те, що наразі існує дуже багато плагінів до нього, одним з най популярніших є плагін з мінімаліьним графічним інтерфейсом користувач, який облегшує роботу з файлофою системою серфера через наглядність (мал.5.11.).



Мал.5.11. Приклад використання плагіну з графічним інтерфейсом для Termius.

Також на момент початку розробки, окрім з'єднання по SSH, Termius надавав змогу безкоштовного користування для з’єднання по протоколу SFTP, який наразі став платним.

SFTP (SSH File Transfer Protocol або ж – протокол передачі файлів за SSH) – це захищений протокол передачі файлів, який працює над протоколом SSH. Він забезпечує повну безпеки та аутентифікацію SSH.

SFTP в значній мірі замінив застарілий FTP як протокол передачі файлів і швидко замінює FTP / S. Він надає всі функціональні можливості,

пропоновані цими протоколами, але надійніше і з легшою конфігурацією. Завдяки йому немає причин більше використовувати застарілі протоколи.

Цей протокол захищає цілісність даних за допомогою шифрувальних та криптографічних хеш-функцій та аутентифікує як сервер, так і користувача.

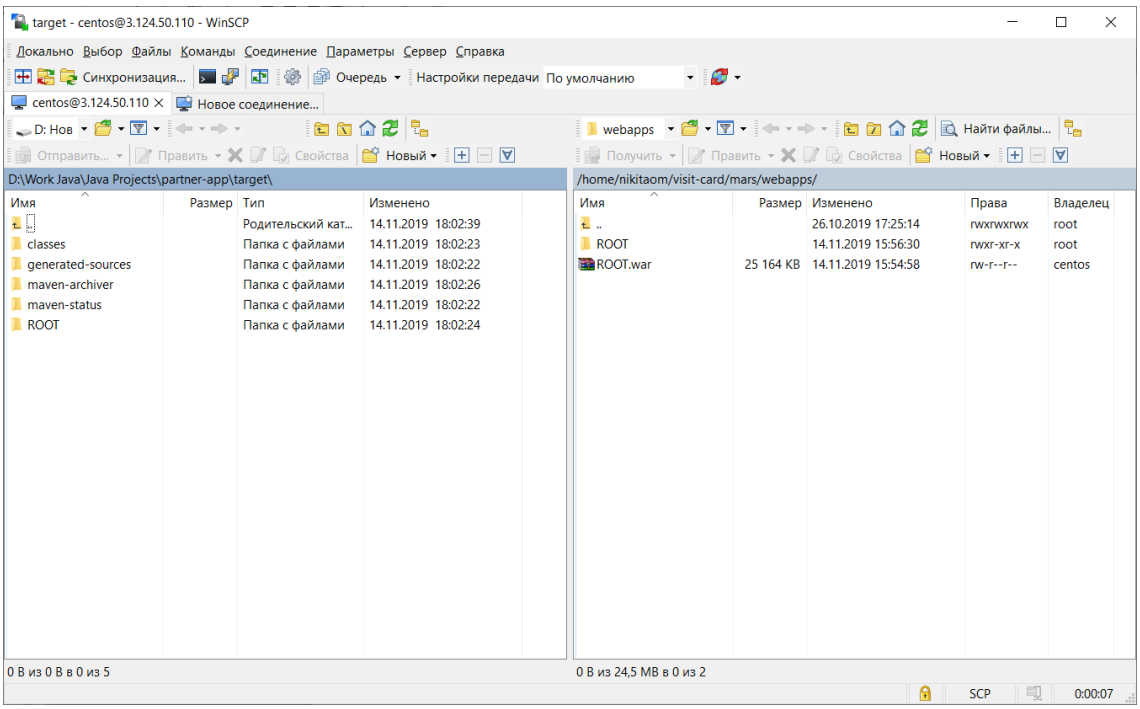
Тому на додачу до Termius, було обрано програму для з’єднання з віддаленими серверами та передачі файлів по SFTP – WinSCP.

WinSCP - це графічний кліент SFTP (протокол передачі файлів SSH) для Windows з відкритим исходним кодом. Він також підтримує протокол SCP (протокол захищеної копії). Попередньо призначені для захищеного копіювання файлів між комп'ютерами та серверами, підтримуючи ці протоколи.

Приклад інтерфейсу даного ПЗ можна побвчити на мал.5.12., де пеаралельно відкрито файловий менеджер локального ПК (ліворуч), та файловий менеджер віддаленого сервера (праворуч).

Звичайним перетаскуванням файлів/папок з ПК на сервер або ж навпаки можна ініціювати операцію копіювання даних.

WinSCP підтримує роботу з декількома серверами одночасно забезпечуючи це відкриттям з’єднання по SFTP з кожним новим сервером у новій вкладці.



Мал.5.12. Паралельно відкриті файлові менеджери локального ПК та серверу.

ВИСНОВОК ДО РОЗДІЛУ 5

За для забезпечення постійного та відкритого доступу до web-додатку, та змоги користуватися сервісом електронних візиток та додатком на Android, вонибули розміщені у всесвітній мережі Internet.

На даний момент будь-хто бажаючий може скористатись послугами даного сервісу, відвідавши відповідні ресурси та/або завантаживши программу.

ВИСНОВКИ

При написанні дипломного проекту, було створено сервіс електронних візитних карток, а саме web-додаток та Android-додаток, який є доповненням (клієнтом) web-додатку. Даний сервіс було створення за для вирішенні проблем, що виникають у людини при користуванні візитними картками у тому вигляді що ми маємо (фізичними). Адже користуючись даними додатками, немає необхідності користуватися фізичними візитними картками, витрачати папір, а з цього виходить що не так багато дерев буде вирубано за для їх виготовлення, зникає необхідність у пошкук місця для зберігання та пошуку карток при необхідності. А завдяки зручному та зрозумілому інтерфейсу додатків, будь-хто зможе з легкістю створити, або отримати та корисуватися електронною візиткою.

Даний сервіс електронних візитних карток, має можливість стати популярним, через декілька, але дуже вагомих причин, перше – це те що він в рази спрощує життя, замінюючи купу паперу, на один додаток у телефоні, або взагалі на посилання, адже зараз йде час диджиталізації інформації, а друге – це гнучкість керування, свтрворюваними візитними картками, так як, наразі зміни якоїсь контрактної інформації про власниеа картки, в паперовом варіанті картки вона ніколи не змінится, і людина що має цю картку може тривалий чам мати її при собі, розразовуючи на те, що в неї є контактна їнформація про людину/організацію, чиєю карткою вона володіє, а насправді вона вже буде застарілою. У представленому додатку, зміна будь-якої їнформації на кардці – це лише справа кількох хвилин.

Представлений сервіс має великі перспективи у майбутньому, зокрема приносити прибуток, завдяки розміщенню реклами, а в основному через добавлення платної версії, в якій створення картки буде платним. Та незважаючи на це, ціна буде менша за друк візитних карток на папері.

Наразі сервіс електронних візиток, вуключно з web-додатком та Android-додатком є готовим продуктом, який доступний до використання, але як і будь-який існуючий продукт, до нього можна внести багато змін та додати модулів, для його блільш ефективної, швидкої та гнучкої роботи. В майбутньом планується, доробити персональних кабінет користувача, та змінити спосіб зберігання інформації.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. https://ru.stackoverflow.com
2. https://developer.android.com/training/basics/firstapp
3. https://proselyte.net/tutorials/spring-tutorial-full-version/spring-mvc-framework/
4. https://tproger.ru/translations/building-a-web-app-with-java-servlets/
5. https://www.postgresql.org
6. https://www.w3.org/Protocols/rfc2616/rfc2616-sec9.html
7. https://spring.io
8. http://tomcat.apache.org
9. https://hibernate.org/orm/
10. https://en.wikipedia.org/wiki/Java_(programming_language)
11. https://aws.amazon.com/ru/

ДОДАТОК А

POM.XML

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
<modelVersion>4.0.0</modelVersion>
<groupId>partizan.global</groupId>
    <artifactId>partner-app</artifactId>
    <version>1.0</version>
    <packaging>war</packaging>
    <name>partner-app Maven Webapp</name>
    <properties>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
<maven.compiler.source>1.7</maven.compiler.source>
<maven.compiler.target>1.7</maven.compiler.target>
    </properties>
    <dependencies>
        <dependency>
<groupId>org.springframework</groupId>
            <artifactId>spring-core</artifactId>
<version>5.1.8.RELEASE</version>
        </dependency>
        <dependency>
<groupId>org.springframework</groupId>
            <artifactId>spring-web</artifactId>
 <version>5.1.8.RELEASE</version>
        </dependency>
        <dependency>
<groupId>org.springframework</groupId>
            <artifactId>spring-webmvc</artifactId>

<version>5.1.8.RELEASE</version>
        </dependency>
        <dependency>

<groupId>org.postgresql</groupId>
<artifactId>postgresql</artifactId>
   <version>42.2.5</version>
        </dependency>
        <dependency>
  <groupId>org.hibernate</groupId>
            <artifactId>hibernate-core</artifactId>

<version>5.4.3.Final</version>
        </dependency>

        <dependency>

<groupId>org.hibernate</groupId>
            <artifactId>hibernate-validator</artifactId>

<version>6.0.17.Final</version>
        </dependency>
```

```
        <dependency>
<groupId>org.springframework.data</groupId>
            <artifactId>spring-data-jpa</artifactId>
<version>2.1.5.RELEASE</version>
        </dependency>

        <dependency>
<groupId>javax.servlet</groupId>
<artifactId>javax.servlet-api</artifactId>
<version>4.0.0</version>
<scope>provided</scope>
        </dependency>

        <dependency>
<groupId>javax.servlet.jsp</groupId>
<artifactId>javax.servlet.jsp-api</artifactId>
<version>2.3.1</version>
<scope>provided</scope>
        </dependency>

        <dependency>
<groupId>javax.persistence</groupId>
<artifactId>javax.persistence-api</artifactId>
            <version>2.2</version>
        </dependency>

        <dependency>
<groupId>javax.servlet</groupId>
<artifactId>jstl</artifactId>
            <version>1.2</version>
        </dependency>

        <dependency>
<groupId>net.minidev</groupId>
            <artifactId>json-smart</artifactId>
            <version>2.3</version>
        </dependency>

        <dependency>
            <groupId>commons-fileupload</groupId>
            <artifactId>commons-fileupload</artifactId>
<version>1.3.1</version>
        </dependency>

        <dependency>
<groupId>org.springframework.security</groupId>
            <artifactId>spring-security-web</artifactId>
<version>5.1.2.RELEASE</version>
        </dependency>
        <dependency>
```

```
<groupId>org.springframework.secur
ity</groupId>
            <artifactId>spring-
security-config</artifactId>

<version>5.1.2.RELEASE</version>
        </dependency>
        <dependency>
<groupId>org.apache.velocity</grou
pId>
<artifactId>velocity</artifactId>
            <version>1.7</version>
        </dependency>
    </dependencies>
    <build>
        finalName>ROOT</finalName>
        <plugins>
            <plugin>
                <artifactId>maven-
war-plugin</artifactId>

<version>2.4</version>
                <configuration>

<failOnMissingWebXml>false</failOn
MissingWebXml>
                </configuration>
            </plugin>
            <plugin>

<groupId>org.apache.maven.plugins<
/groupId>
                <artifactId>maven-
compiler-plugin</artifactId>
                <configuration>

<source>1.8</source>
<target>1.8</target>
                </configuration>
            </plugin>
        </plugins>
    </build>
</project>
```

Controllers (user, partner, login).

```
@Controller
public class UserViewController {

    @Autowired
    public UserService
userService;
    @Autowired
    public RoleService
roleService;

    @GetMapping("/new-user")
    public String newUsr(ModelMap
modelMap) {

modelMap.addAttribute("user", new
User());
        return "add-user";
    }

    @PostMapping("/new-user")
    public String newUser(User
user, ModelMap modelMap) {
        for (User
u:userService.findAll()
                ) {

if(u.getEmail().equals(user.getEma
il())){

modelMap.addAttribute("info","User
with such email already exist");
                return
"redirect:/new-user";
            }
        }

user.setRole(roleService.findRoleB
yId(1l));
        userService.save(user);
        return "redirect:/all-
users";
    }

    @GetMapping("/delete-
user/{id}")
    public String
deleteUser(@PathVariable long id)
{

userService.deleteById(id);
        return "redirect:/all-
users";
    }

    @GetMapping("/all-users")
    public String
showAllUsers(ModelMap model) {

model.addAttribute("users",
userService.findAll());
        return "all-users";
    }
}
```

```
@Controller
public class PartnerViewController
{

    @Autowired
    PartnerService partnerService;

    @Autowired
    PartnerControllerService
partnerControllerService;

    @Autowired
    StatisticsService
statisticsService;

    @GetMapping("/")
    public String mainPage() {
        return "index";
    }

    @GetMapping("/new-partner")
    public String
newPartner(ModelMap modelMap) {

modelMap.addAttribute("partner",
new Partner());
        return "add-partner";
    }

    /*upper case hash*/
    @PostMapping("/new-partner")
    public String newUser(Partner
partner) {
        for (Partner p :
partnerService.findAll()
                ) {
            if
(p.getHashKey().equals(partner.get
HashKey().toUpperCase())) {
                return
"redirect:/new-partner";
            }
        }

partner.setHashKey(partner.getHash
Key().toUpperCase());
```

```
partner.setData(partnerControllerS
ervice.getDataJson());

partnerService.save(partner);
        return "redirect:/all-
partners";
    }

    @GetMapping("/delete/{id}")
    public String
delete(@PathVariable long id) {

partnerService.deleteById(id);
        return "redirect:/all-
partners";
    }

    @GetMapping("/all-partners")
    public String showAll(ModelMap
model) {

model.addAttribute("partners",
partnerService.findAll());
        return "all-partners";
    }

    @Transactional
    @GetMapping("/add-info/{id}")
    public String addInfo(ModelMap
modelMap, @PathVariable Long id) {
        Partner partner =
partnerService.findPartnerById(id)
;

modelMap.addAttribute("partner",
partner);

modelMap.addAttribute("companyName
",
partnerControllerService.getCompan
yName(id));

modelMap.addAttribute("companyDesc
",
partnerControllerService.getCompan
yDescription(id));

modelMap.addAttribute("bgColor",
partnerControllerService.getBgColo
r(id));

modelMap.addAttribute("textColor",
partnerControllerService.getTextCo
lor(id));

modelMap.addAttribute("image",
partnerControllerService.getImage(
id));

        Logger logger =
Logger.getLogger(Partner.class.get
Name());
        logger.info("Data = " +
partner.getData());

        return "data-form";
    }

    @Transactional
    @PostMapping("/add-info/{id}")
    public String
addInfo(@RequestBody String data,

@PathVariable Long id) {
        Partner partner =
partnerService.findPartnerById(id)
;

//        if
(partner.getImage().length < 1) {
//            partner.setImage(new
byte[]{(byte) 0xe0, 0x0});
//        }

        Logger logger =
Logger.getLogger(Partner.class.get
Name());
        logger.info("Data got from
jsp = " + data);


        partner.setData(data);


partnerService.save(partner);

        return "redirect:/all-
partners";
    }

    @Transactional
    @GetMapping("/add-image/{id}")
    public String
addImage(ModelMap modelMap,
@PathVariable Long id) {
        Partner partner =
partnerService.findPartnerById(id)
;

modelMap.addAttribute("partner",
partner);
        return "add-image";
    }

    @Transactional
    @PostMapping("/add-
image/{id}")
    public String
addImage(@RequestParam("imageFile"
) MultipartFile imageFile,
/*Partner partner,*/ @PathVariable
```

```
Long id) throws IOException,
SQLException {
        Partner currentPartner =
partnerService.findPartnerById(id)
;

        byte[] byteObjects = new
byte[imageFile.getBytes().length];
        int i = 0;
        for (byte b :
imageFile.getBytes()) {
            byteObjects[i++] = b;
        }

currentPartner.setImage(byteObject
s);

System.out.println(currentPartner.
toString());

partnerService.update(currentPartn
er);

//partnerControllerService.getHtml
(id);

        return "redirect:/add-
info/" + id;
    }

    @Transactional
    @GetMapping("/partner-
page/{hashKey}")
    public String
getPartnerPage(ModelMap modelMap,
@PathVariable String hashKey,
HttpServletRequest request) {
        Partner partner =
partnerService.findPartnerByHashKe
y(hashKey.toUpperCase());
        if (partner != null) {
            String ip =
partnerControllerService.getClient
IpAddress(request);
            Statistics statistics
=
partnerControllerService.getStatis
tics(partner,ip);

statisticsService.save(statistics)
;


modelMap.addAttribute("html",
partnerControllerService.getHtml(h
ashKey.toUpperCase()));


modelMap.addAttribute("name",partn
er.getName());

modelMap.addAttribute("img",partne
rControllerService.getImage(partne
r.getId()));

modelMap.addAttribute("desc",partn
erControllerService.getCompanyDesc
ription(partner.getId()));

modelMap.addAttribute("textColor",
partnerControllerService.getTextCo
lor(partner.getId()));

modelMap.addAttribute("bgColor",pa
rtnerControllerService.getBgColor(
partner.getId()));


            return "partner-page";
        } else {

modelMap.addAttribute("info",
"Partner page not exist");
            return "404-not-
found";
        }
    }

}
@Controller
public class LoginController {

    @Autowired
    public UserService
userService;

    @Autowired
    public RoleService
roleService;

    @Autowired
    private BCryptPasswordEncoder
bCryptPasswordEncoder;

    @GetMapping("/log-in")
    public String login(Model
model) {
        model.addAttribute("user",
new User());
        return "login";
    }

    @PostMapping("/log-in")
    public String login(User
user,ModelMap modelMap) {
        User userEx =
userService.findUserByEmail(user.g
etEmail());
        if(userEx == null){

modelMap.addAttribute("nouser","no
users with such email");
```

```
            return "redirect:/log-
in";
        }
        if(user == null)return
"redirect:/log-in";
        else
if((user.getEmail().equals(userEx.
getEmail())&&

((user.getPassword()).equals(userE
x.getPassword()))))){


modelMap.addAttribute("info","welc
ome "+userEx.getName());
            return "redirect:/main";
        }
        else {

modelMap.addAttribute("pass","inco
rrect password");
            return "redirect:/log-in";
        }
    }

    @GetMapping("/main")
    String main(){
        return "main";
    }
```

SERVICES

```
@Service
public class
PartnerControllerServiceImpl
implements
PartnerControllerService {

    @Autowired
    public PartnerRepository
partnerRepository;

    @Autowired
    public StatisticsService
statisticsService;


    @Override
    public String getHtml(String
hashKey) {

        Partner partner =
partnerRepository.findPartnerByHas
hKey(hashKey);
        String imageBase64 = "";
        String finalHtml = "";
        String htmlBody = "";
        Object obj = null;

        if (partner.getImage() ==
null)
            imageBase64 = " ";
        else
            imageBase64 =
Base64.getEncoder().encodeToString
(partner.getImage());

        String htmlImage =  "<img
src=\"  " +
imageBase64 + "\" />";


        try {
            obj = new
net.minidev.json.parser.JSONParser
().parse(partner.getData());
        } catch
(net.minidev.json.parser.ParseExce
ption e) {
            e.printStackTrace();
        }

        if(obj != null){
        JSONObject jo =
(JSONObject) obj;

        String name = (String)
jo.get("name");
        String description =
(String) jo.get("description");
        String bgColor = (String)
jo.get("bgColor");
        String textColor =
(String) jo.get("textColor");
        String linkColor =
(String) jo.get("textColor");
        String alinkColor =
(String) jo.get("textColor");
        String vlinkColor =
(String) jo.get("textColor");

        htmlBody = "<html>\n <body
bgcolor=\"" + bgColor + "\"
text=\"" + textColor +
                "\" link=\"" +
linkColor + "\" alink=\"" +
alinkColor +
                "\" vlink=\"" +
vlinkColor + "\">\n" +
                "<center>\n" +
                htmlImage+
                "<h1>" + name +
"</h1>\n" + "</br>" +
                description  +
                "</center>\n" +
                "</body>\n
</html>";

        finalHtml = htmlBody;
        }else {
            finalHtml = "<html>\n
<body>" +htmlImage + "<h1>No
information ...</h1></body>\n
</html>";
        }
        return finalHtml;
    }

    @Override
    public String
getPartnerJson(String hashKey) {
        Partner partner =
partnerRepository.findPartnerByHas
hKey(hashKey);
        String imageBase64 = "";

        Object obj = null;
        String partnerJSON;

        if (partner.getImage() ==
null)
            imageBase64 = " ";
        else
            imageBase64 =
```

```
Base64.getEncoder().encodeToString
(partner.getImage());

        String imageBase64Final =
imageBase64;


        try {
            obj = new
net.minidev.json.parser.JSONParser
().parse(partner.getData());
        } catch
(net.minidev.json.parser.ParseExce
ption e) {
            e.printStackTrace();
        }

        if(obj != null){
            JSONObject jo =
(JSONObject) obj;

            String name = (String)
jo.get("name");
            String description =
(String) jo.get("description");
            String bgColor =
(String) jo.get("bgColor");
            String textColor =
(String) jo.get("textColor");

            partnerJSON =
"{\"image\":\""+imageBase64Final+"
\"," +

"\"name\":\"" +name+"\","+

"\"description\":\""
+description+"\","+

"\"textColor\":\""
+textColor+"\","+

"\"bgColor\":\"" +bgColor+"\"}";




        }else {
            partnerJSON =
"{\"image\":\""+imageBase64Final+"
\"," +
                    "\"name\":\""
+"No name"+"\","+

"\"description\":\"" +"No
description"+"\","+

"\"textColor\":\""
+"#ffffff"+"\","+

"\"bgColor\":\"" +"#000000"+"\"}";
        }
        return partnerJSON;
    }

    @Override
    public String getImage(Long
id){
        Partner partner =
partnerRepository.findPartnerById(
id);
        String imageBase64 = "";
        if (partner.getImage() ==
null)
            imageBase64 = " ";
        else
            imageBase64 =
" "+
Base64.getEncoder().encodeToString
(partner.getImage());
        return imageBase64;
    }

    @Override
    public Statistics
getStatistics(Partner
partner,String ip) {
        Statistics statisrtics =
new Statistics();

statisrtics.setTimestamp(new
Date());

statisrtics.setPartner(partner);

statisrtics.setSourceIp(ip);

        return statisrtics;
    }

    @Override
    public  String
getClientIpAddress(HttpServletRequ
est request) {
        String xForwardedForHeader
= request.getHeader("X-Forwarded-
For");
        if (xForwardedForHeader ==
null) {
            return
request.getRemoteAddr();
        } else {
            // As of
https://en.wikipedia.org/wiki/X-
Forwarded-For
            // The general format
of the field is: X-Forwarded-For:
client, proxy1, proxy2 ...
            // we only want the
client
            return new
StringTokenizer(xForwardedForHeade
```

```
r, ",").nextToken().trim();
        }
    }

    @Override
    public Integer
getStatForPartner(Long id) {
        int size =
statisticsService.findAllByPartner
Id(id);
        return size;
    }



    @Override
    public String getDataJson() {
        return "{\"name\" : \"Your
Company name\" ," +
                "\"description\" :
\"Your Company name description\"
," +
                "\"bgColor\" :
\"#ffffff\" ," +
                "\"textColor\" :
\"#304754\" }" ;
    }

    @Override
    public String
getCompanyName(Long id) {
        Partner partner =
partnerRepository.findPartnerById(
id);
        Object obj = null;
        String name;
        try {
            obj = new
net.minidev.json.parser.JSONParser
().parse(partner.getData());
        } catch
(net.minidev.json.parser.ParseExce
ption e) {
            e.printStackTrace();
        }

        if(obj != null) {
            JSONObject jo =
(JSONObject) obj;

            name = (String)
jo.get("name");
        } else name = "Company
name";


            return name;
    }

    @Override
    public String
getCompanyDescription(Long id) {
        Partner partner =
partnerRepository.findPartnerById(
id);
        Object obj = null;
        String desc;
        try {
            obj = new
net.minidev.json.parser.JSONParser
().parse(partner.getData());
        } catch
(net.minidev.json.parser.ParseExce
ption e) {
            e.printStackTrace();
        }

        if(obj != null) {
            JSONObject jo =
(JSONObject) obj;

            desc = (String)
jo.get("description");
        } else desc = "No
description1";


        return desc;
    }

    @Override
    public String getBgColor(Long
id) {
        Partner partner =
partnerRepository.findPartnerById(
id);
        Object obj = null;
        String bgColor;
        try {
            obj = new
net.minidev.json.parser.JSONParser
().parse(partner.getData());
        } catch
(net.minidev.json.parser.ParseExce
ption e) {
            e.printStackTrace();
        }

        if(obj != null) {
            JSONObject jo =
(JSONObject) obj;

             bgColor = (String)
jo.get("bgColor");
        } else bgColor =
"#ffffff";


        return bgColor;
    }
```

```
    @Override
    public String
getTextColor(Long id) {
        Partner partner =
partnerRepository.findPartnerById(
id);
        Object obj = null;
        String textColor;
        try {
            obj = new
net.minidev.json.parser.JSONParser
().parse(partner.getData());
        } catch
(net.minidev.json.parser.ParseExce
ption e) {
            e.printStackTrace();
        }
        if(obj != null) {
            JSONObject jo =
(JSONObject) obj;

            textColor = (String)
jo.get("textColor");
        } else textColor =
"#304754";

        return textColor;
    }

}
```

VIEWS (data-form, log-in, all-partners, partner page, add-partner)

```
<%@ page
contentType="text/html;charset=UTF
-8" language="java" %>
<%@ taglib prefix="form"
uri="http://www.springframework.or
g/tags/form" %>
<%@ taglib prefix="c"
uri="http://java.sun.com/jsp/jstl/
core" %>
<%@ taglib prefix="spring"
uri="http://www.springframework.or
g/tags" %>
<%@ taglib prefix="s"
uri="http://www.springframework.or
g/tags" %>

<html>
<head>
    <title>Partners page</title>
    <script
src="https://maxcdn.bootstrapcdn.c
om/bootstrap/3.3.7/js/bootstrap.mi
n.js"></script>
    <link
href="https://maxcdn.bootstrapcdn.
com/bootstrap/3.3.7/css/bootstrap.
min.css" rel="stylesheet"/>
    <!--<script
type="text/javascript"
src="js/main.js"></script></head>-
->
    <link href="<c:url
value="/static/css/style.css"/>"
rel="stylesheet">
<body>

<script type="text/javascript">

    var bodyColor = '#ffffff';
    var textColor="#304754";
    var jsonStr = '';

    function sendJSON(){
    var pref = "{";
    var post = "}";
    var name =
document.getElementById('name').va
lue ;
    var strDesc =
document.getElementById('descripti
on').value;
    strDesc =
strDesc.replace(/"/g,"'");
    bodyColor =
document.getElementById('bodycolor
').value;
    textColor =
document.getElementById('textcolor
').value;

     jsonStr = pref
        + '"' +"name"+'"' + " : "+
'"'+name+ '" ,'
        + '"' +"description"+'"' +
" : "+ '"'+strDesc+ '" ,'

        + '"' +"bgColor"+'"' + " :
"+ '"'+bodyColor+ '" ,'
        + '"' +"textColor" +'"' +
" : "+ '"'+textColor+ '" '
        + post;


        xhr = new
XMLHttpRequest();
        var ip =
window.location.hostname;

        /*EDIT PORT for ENGINX*/
        var url =
"http://"+ip+"/add-
info/${partner.id}";
        xhr.open("POST", url,
true);

xhr.setRequestHeader("Content-
type", "application/octet-
stream");
        xhr.onreadystatechange =
function () {
            if (xhr.readyState ==
4 && xhr.status == 200) {
                var json =
JSON.parse(xhr.responseText);

            }
        }
        xhr.send(jsonStr);
        console.log(jsonStr);
        console.log(url);


    }


window.addEventListener('load',
function() {

document.querySelector('input[type
="file"]').addEventListener('chang
e', function() {
            if (this.files &&
```

```
this.files[0]) {
                var img =
document.getElementById('myImg');

//document.querySelector('img[2]')
;  // $('img')[0]
                img.src =
URL.createObjectURL(this.files[0])
; // set src to file url
                img.onload =
imageIsLoaded; // optional onload
event listener
            }
        });
    });

    window.onload = function () {
        var fileupload =
document.getElementById("image");
        var filePath =
document.getElementById("spnFilePa
th");
        var image =
document.getElementById("myImg");
        image.onclick = function
() {
            fileupload.click();
        };
        fileupload.onchange =
function () {
            var fileName =
fileupload.value.split('\\')[fileu
pload.value.split('\\').length -
1];
            filePath.innerHTML =
"<b>Selected File: </b>" +
fileName;
        };

        var imgDb1 =
document.getElementById('imgDb1');
        var s =" ${image} ";
        if(s.length < 40) {
            imgDb1.src="<c:url
value="/static/images/YourLogo.png
"/> ";
          //
document.body.appendChild(imgDb1);
            console.log(s);
        }
    };



</script>

<nav role="navigation"
class="navbar ">
    <div class="navbarleft">
        <img
src="/static/images/MAIN-
LOGO.png">
    </div>
    <div class="navbarright">
        <img
src="/static/images/Header_Graphic
s.jpg">
    </div>


</nav><hr>
<div class="mainleftdiv">
    <div id="editForm">
        <p class="titletext">Edit
information</p>
        <div class="">
            <form:form
name="form1" method="post"
modelAttribute="partner"
enctype="multipart/form-data"
action="/add-image/${partner.id}">
                <div style="float:
left">


                <fieldset
class="form-group">
                    <label>Select
your logo:</label><br>
                    <img
id="myImg" src="<c:url
value="/static/images/YourLogo.png
"/>" alt="Image not found ..."

height=200 width=200/>
                    <input
id="image" class="icon-folder-
open" type="file" name="imageFile"
style="display: none"/><br>
                    <span
id="spnFilePath"></span>
                </fieldset>
            </div>
            <div id="imgDb"
style="margin-left: 300px;
padding-bottom: 50px;">
                <div>
                <fieldset
class="form-group">
                    <label>Curent
logo:</label><br>
                    <img
id="imgDb1" src="${image}" alt="No
logo yet ..." height=200
width=200/>
                </fieldset>
                </div>
            </div>
             <button class="btn
add_user_btn" type="submit" >Save
image</button>
            </form:form>
```

```
        <form id="dataForm"
              action="/all-
partners">
            <fieldset class="form-
group">
                <label >Company
name</label>
                <input type="text"
class="form-control input_white"
name="name" id='name'
value=${companyName}><br>
            </fieldset>
            <fieldset class="form-
group">
                <label
>Description</label>
                <textarea
id="description" class="form-
control"
name="description">${companyDesc}<
/textarea><br>
            </fieldset>

            <p
class="white">Choose  colors:</p>
                <div>
                    <input
type="color" id="bodycolor"
name="head" value=${bgColor}>
                    <label
for="bodycolor">Background
color</label>
                </div>

                <div>
                    <input
type="color" id="textcolor"
name="head" value=${textColor}>
                    <label
for="textcolor">Text color</label>
                </div>
            <button
onclick="sendJSON()"class="btn
add_user_btn">SAVE</button>

            <!--
formaction="/all-partners" -->
        </form>
    </div>
</div>
</div>
</body>
</html>

<%@ page
contentType="text/html;charset=UTF
-8" language="java" %>
<%@ taglib prefix="form"
uri="http://www.springframework.or
g/tags/form" %>
<%@ taglib prefix="c"
uri="http://java.sun.com/jsp/jstl/
core" %>

<html>
<head>
    <title>Create-user</title>
    <script
src="https://maxcdn.bootstrapcdn.c
om/bootstrap/3.3.7/js/bootstrap.mi
n.js"></script>
    <link
href="https://maxcdn.bootstrapcdn.
com/bootstrap/3.3.7/css/bootstrap.
min.css" rel="stylesheet"/>
    <link href="<c:url
value="/static/css/style.css"/>"
rel="stylesheet">
</head>
<body>

<style>
    .display-new-chat-windowp
.new-chat-window-inputp {
      background:
url(/static/images/Password-
icon.png) no-repeat scroll 0
center;
    }
    .display-new-chat-windowe
.new-chat-window-inpute {
        background:
url(/static/images/Login-icon.png)
no-repeat scroll 0 center;
    }
    input[type="email"],
input[type="password"],
input[type="text"],
    select.form-control {
        padding-left: 20px;
    }

</style>
<nav role="navigation"
class="navbar ">
    <div class="navbarleft">
        <img
src="/static/images/MAIN-
LOGO.png">
    </div>
    <div class="navbarright">
        <img
src="/static/images/Header_Graphic
s.jpg">
    </div>


</nav><hr>
<div class="container">
    <div class="loginForm">
```

```
        <form action="/log-in"
method="post">
            <div class="form-group
display-new-chat-windowe">
                <i class="fa fa-
search ipad"></i>
                <input
type="email" name="email"
class="form-control new-chat-
window-inpute"
placeholder="Email"
                       required
pattern="[^ @]*@[^ @]*"/>

                <p style="color:
red"> ${nouser}</p>
            </div>
            <div class="form-group
display-new-chat-windowp">


                <i class="fa fa-
search ipad"></i>

                <input
type="password" name="password"
class="form-control form-control
new-chat-window-inputp
input_white"
placeholder="Password"
                       required
pattern="^[a-zA-Z0-9-_\.]{1,20}$">
                <p style="color:
red"> ${nopass}</p>

                <button
type="submit" class="btn
loginbtn">LOGIN</button>
            </div>
        </form>
    </div>
</div>
</body>
</html>


<%@ page
contentType="text/html;charset=UTF
-8" language="java" %>
<%@ taglib prefix="form"
uri="http://www.springframework.or
g/tags/form" %>
<%@ taglib prefix="c"
uri="http://java.sun.com/jsp/jstl/
core" %>



<html>
<head>
    <title>All-partners</title>
    <script
src="https://maxcdn.bootstrapcdn.c
om/bootstrap/3.3.7/js/bootstrap.mi
n.js"></script>
    <link
href="https://maxcdn.bootstrapcdn.
com/bootstrap/3.3.7/css/bootstrap.
min.css" rel="stylesheet"/>
    <link href="<c:url
value="/static/css/style.css"/>"
rel="stylesheet">
</head>
<body>
<nav role="navigation"
class="navbar ">
    <div class="navbarleft">
        <img
src="/static/images/MAIN-
LOGO.png">
    </div>
    <div class="navbarright">
        <img
src="/static/images/Header_Graphic
s.jpg">
    </div>


</nav><hr>

<div class="topbtn">
    <a href="/new-partner"
type="button" class="btn
add_user_btn">+ADD PARTNER</a>
</div>

<div class="partnertable">
    <table id="table" class="table
table-striped  ">
        <thead>
        <tr class="trh">
            <td>Id</td>
            <td>Name</td>
            <td>HashKey</td>
            <td>Add info</td>

            <td>Show .html</td>
            <td
class="column">Delete</td>
        </tr>
        </thead>
        <tbody>
        <c:forEach
items="${partners}" var="partner">
            <tr>
                <td>${partner.id}

<td>${partner.name}</td>

<td>${partner.hashKey}</td>
                <td
class="column">
```

```
                    <a
type="button" class="" href="/add-
info/${partner.id}"><span

class="    glyphicon glyphicon-
pencil"></span></a>
                </td>

                <td
class="column">
                    <a
type="button" class=""
href="/partner-
page/${partner.hashKey}"><span

class="glyphicon glyphicon-eye-
open" ></span></a>
                </td>
                <td
class="column">
                    <a
type="button" class=""
href="/delete/${partner.id}"><span

class="glyphicon glyphicon-
trash"></span></a>
                </td>
            </tr>
        </c:forEach>
        </tbody>
    </table>
</div>


</body>
</html>


<%@ page
contentType="text/html;charset=UTF
-8" language="java" %>
<%@ taglib prefix="form"
uri="http://www.springframework.or
g/tags/form" %>
<%@ taglib prefix="c"
uri="http://java.sun.com/jsp/jstl/
core" %>

<html>
<head>
    <title>Create-partner</title>
    <script
src="https://maxcdn.bootstrapcdn.c
om/bootstrap/3.3.7/js/bootstrap.mi
n.js"></script>
    <link
href="https://maxcdn.bootstrapcdn.
com/bootstrap/3.3.7/css/bootstrap.
min.css" rel="stylesheet"/>
    <link href="<c:url
value="/static/css/style.css"/>"
rel="stylesheet">

</head>
<body>
<nav role="navigation"
class="navbar ">
    <div class="navbarleft">
        <img
src="/static/images/MAIN-
LOGO.png">
    </div>
    <div class="navbarright">
        <img
src="/static/images/Header_Graphic
s.jpg">
    </div>


</nav><hr>

<div class="container">
    <div class="loginForm">
        <form:form method="post"
modelAttribute="partner"
action="/new-partner">
            <form:hidden
path="id"/>
            <fieldset class="form-
group">
                <%--<form:label
path="name">Name</form:label>--%>
                <form:input
path="name" placeholder="Enter
your name" type="text"
class="form-control input_white"
required="true"/>
            </fieldset>
            <fieldset class="form-
group">
                <%--<form:label
path="hashKey">hashKey</form:label
>--%>
                <form:input
path="hashKey"  placeholder="Enter
hashkey" type="text" class="form-
control input_white"
required="true"/>
            </fieldset>
            <fieldset class="form-
group">
            <button type="save"
class="btn
add_user_btn">Save</button>
            <a href="/"
type="button" class="btn
cancle">CANCLE</a>
            </fieldset>
        </form:form>
    </div>
</div>
</div>
```

```
</body>
</html>
```

```
<%@ taglib prefix="c"
uri="http://java.sun.com/jsp/jstl/
core" %>
<%@ page
contentType="text/html;charset=UTF
-8" language="java" %>

<html>
<head>
    <meta name="viewport"
content="width=device-width,
initial-scale=1">
    <style>
        .card {
            box-shadow: 0 4px 8px
0 rgba(0,0,0,0.2);
            transition: 0.3s;
            width: 40%;
            border-radius: 5px;
            background-color:
${bgColor};
           margin-top: 25px;
        }

        .card:hover {
            box-shadow: 0 8px 16px
0 rgba(0,0,0,0.2);
        }

        img {
            border-radius: 5px;
        }

        .container {
            padding: 2px 16px;
        }
    </style>
</head>
<body style="background-color:
${bgColor}; color: ${textColor}">


<center>
<div class="card">
    <img src="${img}" alt="Logo"
style="width:99%">
    <div class="container">
        <h4><b>${name}</b></h4>
        <p>${desc}</p>
    </div>
</div>
</center>
</body>
</html>
```

ДОДАТОК Г

API

```
@RestController
@RequestMapping("/api")
public class PartnerController {

    @Autowired
    private PartnerService
partnerService;

    @Autowired
    private
PartnerControllerService
partnerControllerService;


    @Transactional
    @GetMapping("/partner/{hash}")
    public String
gethtml(@PathVariable String
hash){
        Partner partner =
partnerService.findPartnerByHashKe
y(hash.toUpperCase());
        if(partner!= null){
        String html =
partnerControllerService.getHtml(h
ash.toUpperCase());
           return html ;
        }else {
            throw new
ResourceNotFound();
        }
    }

    @Transactional

@GetMapping("/partnerJSON/{hash}")
    public String
getPartnerJson(@PathVariable
String hash){
        Partner partner =
partnerService.findPartnerByHashKe
y(hash.toUpperCase());
        if(partner!= null){
           return
partnerControllerService.getPartne
rJson(hash.toUpperCase()) ;
        }else {
            return "{}";
        }
    }

    @Transactional
    @GetMapping("file/{hash}")
    public void
downloadPDFResource(
HttpServletResponse response,

@PathVariable String hash) throws
IOException {
        String content;
        Partner partner =
partnerService.findPartnerByHashKe
y(hash.toUpperCase());
        if(partner!= null){
           content =
partnerControllerService.getPartne
rJson(hash.toUpperCase()) ;
        }else {
            content = "Nop";
        }

response.setContentType("text/plai
n");

response.setHeader("Content-
Disposition","attachment;filename=
" +hash+".txt");
        ServletOutputStream out =
response.getOutputStream();
        out.println(content);
        out.flush();
        out.close();

        }
    }
```